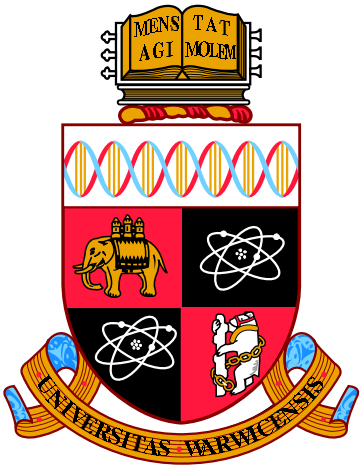


Getting The Most From Medical VOC Data Using Bayesian Feature Learning

by

James Raymond Skinner

Thesis

Submitted to the University of Warwick

for the degree of

Doctor of Philosophy

Centre for Complexity Science

July 2019

THE UNIVERSITY OF WARWICK

Contents

List of Tables

List of Figures

Acknowledgments

I would like to thank my supervisor Rich Savage, who has been a constant source of encouragement and positivity. He has always been generous with his time, and his support and guidance has been invaluable and appreciated, all the way from big-picture research directions down to minor typographical suggestions.

I would also like to thank my second supervisor, James Covington, who has been able to coordinate countless collaborations between academia, industry and medicine, producing rich sources of artificial olfaction data. Thank you James Covington and Simon Spencer for your expert opinions on the relevant parts of my thesis. Thank you to the many people involved in the studies into artificial olfaction performed in this thesis:

There are also very many fellow doctoral students and friends I wish to thank. Thank you to fellow members of "Team Savage": Iliana Peneva, Ayman Boustati, Ale Avalos, Matthew Neal, Katherine Lloyd and Nadia Jankovicova. Thank you to the Warwick Machine Learning Reading Group attendees and presenters: Rob Eyre, Michael Pearce, Jevgenji Gamper, Helen Kochkina, Leo Souliotis, and many others. Thank you to my 2013 MSc/PhD intake: Jason Lewis, Jeremy Reizenstein, Alex Bishop, Conor Finn, Jess Talbot, Janis Klaise, and all the Erasmus Mundus. Thank you for the bouldering squad: Jack Binysh, Jon Skipp, Sami Al-Izzi and Neil Jenkins. Thank you to everybody in the Centre for Complexity Science, especially the hard-working admin staff Heather Robson and Debbie Walker. Thank you Siavash Esfahani for the diabetes collaboration, despite the negative result. Thank you to the Eastleigh crew Mike, Adam, John, Lianne, Jessie, Emily, and Nicole. Thank you to the Southampton crew Tom, Josh, Rhys, Pete, Neal, and Tony. Thank

you to the Leamington crew Alonso, Michael and Steph. Thank you to the pub quiz crew Katherine, John, Rob, Cameron and Vic. Thank you to the D&D crew Joe, Gina, Chris, Henry, Jessie and Matt. Thank you to Becky, Amber, Alex, Rose, George, Ewen, David, Beth, and everybody at the Hampshire Bowman. Your presentations, discussion, enthusiasm, examples, tea-breaks, late nights, discussions, arguments and complaining sessions have helped me more than you know.

Thank you to my wonderful partner Iliana, who has shown more patience than I could ever ask for. Finally, thank you to my Dad and Grandma for your unwavering belief and financial support, without which I would not be writing this thesis. I do not expect you to read it.

Declarations

This thesis is submitted to the University of Warwick in support of my application for the degree of Doctor of Philosophy. It has been composed by myself and has not been submitted in any previous application for any degree. This thesis has not been submitted for a degree at another university.

The work presented (including data generated and data analysis) was carried out by the author except in the following cases. A number of data analyses are described in **Section 2.3**; with the exception of **Section 2.3.1**, the data was collected by collaborators.

Parts of this Thesis have been published by the author. The full list of publiations including material in this thesis is:

1. Parts of **Section 2.3.2** have been published in [Arasaradnam et al., 2016b].
2. Parts of **Section 2.3.3** have been published in [Sahota et al., 2016].
3. Parts of **Section 2.3.4** have been published in [Arasaradnam et al., 2016a].
4. Parts of **Section 2.3.5** have been published in [Esfahani et al., 2015].

Abstract

The metabolic processes in the body naturally produce a diverse set of Volatile Organic Compounds (VOCs), which are excreted in breath, urine, stool and other biological samples. The VOCs produced are odorous and influenced by disease, meaning olfaction can provide information on a person's disease state.

A variety of instruments exist for performing "artificial olfaction": measuring a sample, such as patient breath, and producing a high dimensional output representing the odour. Such instruments may be paired with machine learning techniques to identify properties of interest, such as the presence of a given disease.

Research shows good disease-predictive ability of artificial olfaction instrumentation. However, the statistical methods employed are typically off-the-shelf, and do not take advantage of prior knowledge of the structure of the high dimensional data. Since sample sizes are also typically small, this can lead to suboptimal results due to a poorly-learned model.

In this thesis we explore ways to get more out of artificial olfaction data. We perform statistical analyses in a medical setting, investigating disease diagnosis from breath, urine and vaginal swab measurements, and illustrating both successful identification and failure cases. We then introduce two new latent variable models constructed for dimension reduction of artificial olfaction data, but which are widely applicable. These models place a Gaussian Process (GP) prior on the mapping from latent variables to observations. Specifying a covariance function for the GP prior is an intuitive way for a user to describe their prior knowledge of the data covariance structure. We also enable an approximate posterior and marginal likelihood to be computed, and introduce a sparse variant. Both models have been made available in the R package `stpca` hosted at `https://github.com/JimSkinner/stpca`. In experiments with artificial olfaction data, these models outperform standard feature learning methods in a predictive pipeline.

Abbreviations

AUC	Area Under the ROC Curve
BV	Bacterial Vaginosis
CD	Chrohn's Disease
CI	Confidence Interval
CV	Cross-validation
GBS	Group B Streptococcal Infection
GC	Gas Chromatography
GC-IMS	Gas Chromatography-Ion Mobility Spectrometer
GC-MS	Gas Chromatography-Mass Spectrometer
GP	Gaussian Process
GPC	Gaussian Process Classifier
HE	Hepatic Encephalopathy
IBD	Inflammatory Bowel Disease (IBD)
ICA	Independent Component Analysis
IMS	Ion Mobility Spectrometer
MAP	maximum-a-posteriori
MS	Mass Spectrometer
PCA	Principal Component Analysis
PPCA	Probabilistic Principal Component Analysis
RF	Random Forest
ROC curve	Receiver Operating Characteristic curve
RPCA	Robust Principal Component Analysis
SPCA	Sparse Principal Component Analysis

SpStPCA	Sparse Structured Principal Component Analysis
StPCA	Structured Principal Component Analysis
SVD	Singular Value Decomposition
SVM	Support Vector Machine
TB	Tuberculosis
UC	Ulcerative Colitis
VOC	Volatile Organic Compoind

Notation

Matrices, vectors and scalars are respectively denoted as bold capital (e.g., $\mathbf{X}$), bold lowercase (e.g., $\mathbf{x}$), and non-bold lowercase (e.g., x). Vectors are always column vectors, except when transposed (e.g., $\mathbf{x}^\top$). We often refer to a single column or row of a matrix using the indexed vector notation with the same letter; e.g., $\mathbf{x}_i$ may denote the ith row of the matrix $\mathbf{X}$. It will be made clear whether the index refers to a row or a column. We also refer to single elements of a matrix, in which case we use lowercase capital with subscript indices; e.g., X_{ij} would refer to the element in $\mathbf{X}$ at the ith row and jth column. The characters i, j, l, m are reserved as indices, so will not be used as constants.

The constants n, d, k respectively refer to the number of samples, the dimensionality of a dataset, and the latent dimensionality in a feature learning method. $\mathbf{X}$ will refer to a set of data with dimensions $n \times d$. $\mathbf{x}_i$ is the ith sample, which is also the ith row of $\mathbf{X}$. $\mathbf{V}$ is an $n \times k$ matrix of latent representations. $\mathbf{W}$ is a $d \times k$ "loadings" matrix which takes a sample in the latent space and projects it into the raw measurement space (e.g., $\mathbf{W}\mathbf{v}_i$ projects the latent representation of the ith sample into d dimensions).

Chapter 1

Introduction

This introductory chapter covers background material for this thesis, and is split into three sections.

Section 1.1 introduces the *latent variable model*, and describes a number of well known latent variable models under a consistent syntax. A focus is put on linear latent variable models, but nonlinear models are included to illustrate how non-linearity may be achieved. Latent variable models are an important tool in understanding unsupervised machine learning techniques and how different techniques are connected.

Section 1.2 discusses the statistical background used in this thesis, which is primarily Bayesian statistics. Model selection in a Bayesian setting is discussed, and the reader is introduced to the Laplace approximation for approximating the model evidence. The Empirical Bayes approximation for hyper-parameter selection is then discussed, and it is shown how selecting the hyper-parameters maximising the model evidence is an approximation to full Bayesian inference.

Section 1.3 introduces the Gaussian Process as a distribution over functions with nice analytic properties. When using this as a prior over some unknown latent function and collecting noisy observations at a finite set of points, a posterior can be computed in closed form. A Gaussian process requires a covariance function, which controls the characteristic properties of the functions considered likely, such as smoothness and the range of correlations. These covariance functions are often interpretable, and a list of well known covariance functions is given in **Section 1.3.2**.

1.1 Latent Variable Models

Modern data science often requires working with high dimensional datasets, frequently with more dimensions than observations ($n \ll d$). Reducing the number of dimensions the data are represented in is often desirable; this may aid data exploration through visualisation, improve the predictive accuracy of a classification pipeline or reduce computational requirements. A very simple way of reducing dimensionality is to consider a subset of the variables at hand, but this will often discard relevant information. Instead one may preserve more signal by deriving features from the entirety of the original set of measurements. There are many ways of achieving this.

Latent variable models are an important tool when deriving feature learning techniques. These models consider a dataset as a collection of observations which are arrived at by transforming, and possibly adding noise to, some unobserved latent representation. This latent representation is typically of lower dimension than the observation, and a feature learning technique attempts to infer the latent variables. If the model is representative of reality, the inferred latent variable should capture useful information in lower dimension.

In this work we are particularly interested in linear latent variable models. Here the observations $\mathbf{x}$ are modelled as an affine transformation of the latent variables $\mathbf{v}$:

$$\mathbf{x} \approx \mathbf{W}\mathbf{v} + \boldsymbol{\mu} \tag{1.1}$$

where $\mathbf{W}$ and $\boldsymbol{\mu}$ are fixed parameters, where these are learned differently depending on precise model. Note that only the latent-to-observed mapping is linear; in some linear latent variable models the mapping from observation to reconstruction $\mathbf{W}\mathbf{v}+\boldsymbol{\mu}$ is non-linear. For example, in Dictionary Learning (introduced later in this chapter), finding $\mathbf{v}$ for a given $\mathbf{x}$ and fixed parameters requires solving a non-linear optimisation, so the mapping from $\mathbf{x}$ to $\mathbf{W}\mathbf{v} + \boldsymbol{\mu}$ does not have a liear form that holds across all values of $\mathbf{x}$. Certain models such as PCA and Probabilistic PCA do learn a linear mapping, simplifying the mathematics and interpretation.

It is usually the case that $\boldsymbol{\mu}$ is computed as the mean of each sample. This means that, to clean up notation, we can assume that the data has had the mean subtracted from it, and that $\boldsymbol{\mu} = 0$. It will be assumed in this thesis that data are zero-mean, unless otherwise specified.

Linear latent variable models can be appropriate in the $n \ll d$ setting because there may be insufficient information in the data to learn complex non-linear relationships. The simplicity of the linear model provides additional advantages;

inspecting a single column of $\mathbf{W}$ shows how much each observed dimension contributes to a given latent dimension. Analytic tractability also often leads to simple inference procedures.

A set of well known latent variable models are presented. Many more techniques exist; those presented were chosen to illustrate key ways in which models may differ.

Notation

We use the notation throughout the thesis that n is the number of samples, d is the number of features, and the $n \times d$ matrix of data $\mathbf{X}$ has been centred so that every column has zero mean. The ith sample, which is also the ith row of $\mathbf{X}$, is denoted $\mathbf{x}_i$. $\mathbf{V}$ is the $n \times k$ matrix of latent representations, where $k < d$ is the latent dimensionality. The ith row of $\mathbf{V}$, which is the latent representation of $\mathbf{x}_i$, is denoted $\mathbf{v}_i$. Since we have assumed zero mean data, $\boldsymbol{\mu}$ is taken to be zero unless stated otherwise. The loadings matrix $\mathbf{W}$ is the $d \times k$ matrix which maps the latent space to the space of observations.

1.1.1 Models

Principal Component Analysis

Principal Component Analysis (PCA) [e.g., Bishop, 2006, Section 12.1] is a common technique in data analysis. Given a matrix of data $\mathbf{X} \in \mathbb{R}^{n \times d}$, PCA can be used to compute the first Principal Component (PC), which is a d dimensional vector giving the direction in which the data show the greatest variance. PCs higher than the first are defined as the direction of maximal variance of the data subject to being orthogonal to all previous PCs. There are $\min(d, n)$ principal components that can be computed; this is intuitive since the data cannot vary in more than $\min(d, n)$ orthogonal directions.

A common use of PCA is to reduce the dimensionality of the data. By using the first k PCs as an under-complete basis for the data, a k-dimensional latent representation of each sample is given. A point in the latent space can always be projected back to the d-dimensional observed data space; doing so will place the reconstructions on a k-dimensional linear subspace of the observed data space.

In PCA, one performs the eigen-decomposition of the sample covariance matrix for centred $\mathbf{X}$:

$$\frac{1}{n}\mathbf{X}^\top \mathbf{X} = \mathbf{Q}\boldsymbol{\Lambda}\mathbf{Q}^\top \tag{1.2}$$

where $\mathbf{Q}$ contains the principal components in columns, and the diagonal matrix $\mathbf{\Lambda}$ contains the associated eigenvalues. It is common to consider only the first $k < d$ principal components contained in the columns of the $d \times k$ matrix $\mathbf{Q}_k$, paired with the diagonal $k \times k$ matrix $\mathbf{\Lambda}$ containing the associated eigenvalues. Using these, one can obtain a k-dimensional latent representation of the data

$$\mathbf{V} = \mathbf{X}\mathbf{Q}_k\mathbf{\Lambda}_k^{-\frac{1}{2}} = \mathbf{X}(\mathbf{W}^+)^\top \tag{1.3}$$

where we have defined the linear mapping

$$\mathbf{W}^+ := \mathbf{\Lambda}_k^{-\frac{1}{2}}\mathbf{Q}_k^\top \in \mathbb{R}^{k\times d} \tag{1.4}$$

Under this definition, $\mathbf{W}^{+\top}$ takes a point in the observed data space and maps it to the latent space. $\mathbf{Q}_k$ maps the d-dimensional points to the k-dimensional principal subspace, and $\mathbf{\Lambda}_k^{-\frac{1}{2}}$ re-scales such that each latent variable has variance 1.

Given a point in the latent space, one can always map back to the observed data space through

$$\mathbf{W} := \mathbf{Q}_k\mathbf{\Lambda}_k^{\frac{1}{2}} \in \mathbb{R}^{d\times k} \tag{1.5}$$

$\mathbf{W}^+$ in **Equation 1.4** is the *left inverse* of $\mathbf{W}$ since $\mathbf{W}^+\mathbf{W} = I_k$, but $\mathbf{W}\mathbf{W}^+$ has rank k so cannot equal I_d. This means that we can map from the latent space to the observed space and back without information loss, but the converse is not true unless $k = d$. By mapping observations to the latent space and back, one obtains an approximate reconstruction of the data:

$$\mathbf{X}_k = \mathbf{V}\mathbf{W} = \mathbf{X}\mathbf{Q}_k\mathbf{Q}_k^\top \tag{1.6}$$

where $\mathbf{X}_k$ is a rank-k reconstruction of the original data $\mathbf{X}$, and is the "best" linear rank-k reconstruction in terms of ℓ_2 norm, i.e.,

$$\mathbf{X}_k = \arg\min_{\tilde{\mathbf{X}}} ||\mathbf{X} - \tilde{\mathbf{X}}||_{\mathrm{F}}^2 \quad \text{subject to} \quad \mathrm{rank}(\tilde{\mathbf{X}}) = k \tag{1.7}$$

where $||\mathbf{A}||_{\mathrm{F}} = \sqrt{\sum_i |\mathbf{A}_i|_2} = \sqrt{\sum_{i,j} A_{ij}^2}$ is the Frobenius norm.

PCA is closely related to the Singular Value Decomposition (SVD) of $\mathbf{X}$, which illustrates some of the ideas above more clearly. Performing the SVD of $\mathbf{X}$, we get

$$\mathbf{X} = \bar{\mathbf{U}}\bar{\mathbf{\Sigma}}\bar{\mathbf{V}}^\top \tag{1.8}$$

The standard notation for the SVD does not include the over-bar; we have included

this as a reminder of which variables have been obtained from the SVD of $\mathbf{X}$, and to distinguish these from previously defined variables (such as the latent variables $\mathbf{V}$). Using **Equation 1.8**, one can reformulate the sample covariance matrix:

$$\frac{1}{n}\mathbf{X}^\top\mathbf{X} = \bar{\mathbf{V}}(\frac{1}{n}\bar{\mathbf{\Sigma}}\bar{\mathbf{\Sigma}}^\top)\bar{\mathbf{V}}^\top \tag{1.9}$$

This has a similar form to the sample covariance matrix eigen-decomposition, which is not coincidental. $\bar{\mathbf{V}}$ is orthogonal and $\frac{1}{n}\bar{\mathbf{\Sigma}}\bar{\mathbf{\Sigma}}^\top$ is diagonal, so PCA and SVD have the correspondence:

$$\mathbf{Q}_k = \bar{\mathbf{V}}_k \tag{1.10}$$

$$\mathbf{\Lambda}_k = \frac{1}{n}\bar{\mathbf{\Sigma}}_k\bar{\mathbf{\Sigma}}_k \tag{1.11}$$

$$\mathbf{W} = n^{-\frac{1}{2}}\bar{\mathbf{V}}_k\bar{\mathbf{\Sigma}}_k \tag{1.12}$$

$$\mathbf{W}^+ = n^{\frac{1}{2}}\bar{\mathbf{\Sigma}}_k^{-1}\bar{\mathbf{V}}_k^\top \tag{1.13}$$

$$\mathbf{V} = n^{-\frac{1}{2}}\bar{\mathbf{U}}_k \tag{1.14}$$

This is convenient since performing the SVD of the $n \times d$ matrix $\mathbf{X}$ is potentially computationally faster than computing the full eigen-decomposition of the $d \times d$ sample covariance matrix.

Probabilistic PCA

Probabilistic PCA [Tipping and Bishop, 1999] is a probabilistic latent variable model closely related to PCA. The plate diagram is given in **Figure 1.1** and the probability model is:

$$\mathbf{v}_i \sim \mathcal{N}(\mathbf{0}, I) \tag{1.15}$$

$$\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2 \sim \mathcal{N}(\mathbf{W}\mathbf{v}_i, \sigma^2 I) \tag{1.16}$$

with uniform priors over $\mathbf{W}, \sigma^2$. These constant over the entire domain, so are improper.

Equation 1.16 tells us that given a latent representation, we obtain a distribution over observations centred at $\mathbf{W}\mathbf{v}_i$ with isotropic variance σ^2. Larger σ^2 relates to a noisier modelled data generating process. Since we cannot observe the latent variable, it is of interest to integrate out $\mathbf{v}_i$ to obtain the likelihood [Tipping and Bishop, 1999]:

$$\mathbf{x}_i|\mathbf{W}, \sigma^2 \sim \mathcal{N}(\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \sigma^2 I) \tag{1.17}$$

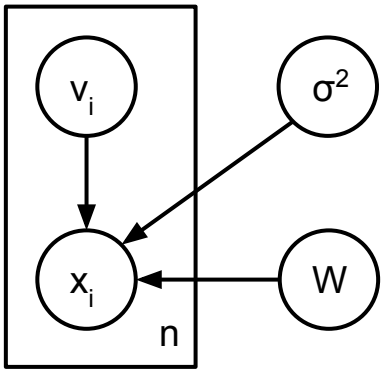


Figure 1.1: *Plate diagram for PPCA, equates to the factorisation* $p(\mathbf{X}, \mathbf{V}, \mathbf{W}, \sigma^2) = \left[\prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{v}_i, \sigma^2, \mathbf{W})p(\mathbf{v}_i)\right] p(\sigma^2)p(\mathbf{W})$.

We can see now that if we do not know $\mathbf{v}_i$, we expect a distribution of observations that, if σ^2 is small, is approximately a low-rank Gaussian.

We can obtain the parameters $\mathbf{W}$, σ^2 by maximising the likelihood of our data. In PPCA this can be done in closed form [Tipping and Bishop, 1999]. This uses the eigen-decomposition of the sample covariance matrix $\frac{1}{n}\mathbf{X}^\top\mathbf{X} = \mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$, $\mathbf{\Lambda} = \text{diag}[\lambda_1, \cdots, \lambda_d]$, giving the Maximum-Likelihood parameters:

$$\mathbf{W}_{\text{ML}} = \mathbf{Q}_k(\mathbf{\Lambda}_k - \sigma^2_{\text{ML}} I)^{\frac{1}{2}}\mathbf{R} \tag{1.18}$$

$$\sigma^2_{\text{ML}} = \frac{1}{d-k}\sum_{i=k+1}^{d} \lambda_k \tag{1.19}$$

Here, $\mathbf{Q}_k$ is the $d \times k$ matrix formed from the first k columns of $\mathbf{Q}$, and $\mathbf{\Lambda}_k = \text{diag}[\lambda_1, \cdots, \lambda_k]$. $\mathbf{R} \in \mathbb{R}^{k\times k}$ is an arbitrary orthonormal matrix representing a non-identifiability in the model, which we may simply pick to be $\mathbf{R} = I$. Computationally, the eigendecomposition can be obtained efficiently from the SVD of $\mathbf{X}$.

One may also be interested in the stochastic mapping from observed to latent space, which is also available in closed form [Tipping and Bishop, 1999]. This uses the definition $\mathbf{M} := \mathbf{W}^\top\mathbf{W} + \sigma^2 I$. Note this is $k \times k$, unlike the similar looking covariance matrix of the likelihood. The mapping is

$$\mathbf{v}_i|\mathbf{x}_i, \mathbf{W}, \sigma^2 \sim \mathcal{N}\left(\mathbf{M}^{-1}\mathbf{W}^\top\mathbf{x}_i, \sigma^2\mathbf{M}^{-1}\right) \tag{1.20}$$

This is centred at a linear transformation of $\mathbf{x}_i$, so both the latent-to-observed (**Equation 1.16**) and observed-to-latent mappings are linear in expectation. One can also see that the uncertainty is the same over all latent representations ($\sigma^2\mathbf{M}^{-1}$),

regardless of $\mathbf{x}_i$.

PPCA recovers PCA exactly in the case of $\sigma^2 \searrow 0$ and $\mathbf{R} = I$. Considering the formula for the $\mathbf{W}_{\text{ML}}$, we can see that in this case, $\mathbf{W}_{\text{ML}} = \mathbf{Q}_k \mathbf{\Lambda}_k^{\frac{1}{2}}$. This exactly recovers the formula for the loadings in PCA, **Equation 1.5**. Considering now the mapping to the latent space **Equation 1.20**, this collapses to a point mass at $(\mathbf{W}^\top \mathbf{W})^{-1} \mathbf{W}^\top$. This is the left-inverse of $\mathbf{W}$, so corresponds exactly to the latent mapping in PCA, **Equation 1.3**.

Whilst the maximum likelihood parameters are available in closed form as above, these can also be computed using Expectation Maximization. This is an iterative method of MAP inference, which is discussed in more detail in **Section 1.1.2**.

Factor Analysis

Factor Analysis (FA) is closely related to PPCA, with the model taking the following form

$$p(\mathbf{x}|\mathbf{v}, \theta) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{v}, \mathbf{\Psi}) \tag{1.21}$$

$$p(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\mathbf{0}, I) \tag{1.22}$$

The only modelling difference to PPCA is that in FA the $d \times d$ matrix $\mathbf{\Psi}$ is diagonal with each element a free parameter, whereas PPCA uses $\sigma^2 I_d$. It is tractable to integrate out the latent variables, arriving at the FA likelihood:

$$p(\mathbf{x}|\theta) = \mathcal{N}(\mathbf{x}|\mathbf{0}, \mathbf{W}\mathbf{W}^\top + \mathbf{\Psi}) \tag{1.23}$$

Comparing this to the PPCA likelihood **Equation 1.17**, it can be seen that FA generalises PPCA, since restricting $\mathbf{\Psi} = \sigma^2 I$ recovers PPCA exactly.

As with PPCA, parameters are determined via maximum likelihood. However, FA does not have a closed form maximum likelihood solution for $\mathbf{\Psi}$ or $\mathbf{W}$. Inference can instead be performed iteratively with Expectation Maximisation [Rubin and Thayer, 1982].

Categorical PCA

So far every method presented is defined for real-valued observations. One can consider modifications to the probability model of PPCA to arrive at PPCA-like algorithms for data in other domains. Murphy [2012, Section 12.4] describes a method for categorical data, where each element of a single sample $\mathbf{x}$ can take one of c values: $x_i \in \{1, \cdots, c\}$, and the values have no meaningful ordering. This would

apply to data such as blood type, which may take a single value from A, B, AB or O.

To arrive at categorical PCA from PPCA, the Gaussian distribution used in the mapping from the latent to observation space is replaced with a categorical distribution. This produces the probability model:

$$p(\mathbf{x}|\mathbf{v}, \boldsymbol{\theta}) = \prod_{i=1}^{d} \text{Cat}(x_i|\sigma(\mathbf{W}_i\mathbf{v} + \mathbf{w}_{0i})) \tag{1.24}$$

$$p(\mathbf{v}) = \mathcal{N}(\mathbf{v}|\mathbf{0}, I) \tag{1.25}$$

$\boldsymbol{\theta} = \{\mathbf{W}_i, \mathbf{w}_{0i}\}_{i=1}^{d}$ contains the parameters of the model, and $\sigma : \mathbb{R}^d \to [0, 1]^d$ is the softmax function

$$\sigma(\mathbf{z}) = \frac{1}{\sum_{j=1}^{d} \exp(z_j)} \begin{bmatrix} \exp(z_1) \\ \vdots \\ \exp(z_d) \end{bmatrix}$$

This "squashes" a vector $\mathbf{z} \in \mathbb{R}^d$ into a vector of positive real values that sum to one. The resultant vector thus parametrises the event probabilities of a d-dimensional categorical distribution.

Unlike PPCA, Categorical PCA does not admit a closed form likelihood. Khan et al. [2010] give a method of approximate inference. This method supports a latent-to-observed mapping made up of a product of Gaussian and Categorical distributions, enabling mixed-type data to be used.

Independent Component Analysis

Independent Component Analysis (ICA) [Hyvrinen and Oja, 2000] is a linear latent variable model in which $\mathbf{W}$ is learned by maximising the statistical independence of the latent dimensions. This is different to searching for uncorrelated features as in PCA, and complete independence is not always achievable with a linear projection. $\mathbf{V}$ and $\mathbf{W}$ are known as the *source* and *mixing* matrices respectively.

The key difference from PCA is that each latent variable v_i is modelled as non-Gaussian and statistically *independent*:

$$p(v_1, \cdots, v_k) = \prod_{i=1}^{k} p(v_i) \tag{1.26}$$

When performing inference, both the source and mixing matrices are inferred, and the columns of the source matrix are inferred to be as close to statistically inde-

pendent as a linear transformation will allow.

Statistical independence between the latent variables cannot be verified without access to the data generating distribution. A proxy for independence must thus be used, leading to a number of different ICA models. A common proxy is non-Gaussianity of the latent variables, with the following rationale: assuming the true source distributions are independent and non-Gaussian, the Central Limit Theorem tells us that an additive combination of these sources will be closer to Gaussian distributed than the individual sources. Decomposing $\mathbf{X}$ into $\mathbf{V}\mathbf{W}^\top$ with maximally non-Gaussian columns of $\mathbf{V}$ may thus un-mix the independent signals.

A number of methods are used to measure non-Gaussianity, which are reviewed by Hyvrinen and Oja [2000] and briefly discussed here. The kurtosis of a Gaussian random variable is zero, so the absolute value or squared value of the empirical kurtosis of each latent variable may be used to measure non-Gaussianity. Kurtosis as a measurement is simple analytically and computationally, but may be sensitive to outliers. Additionally, using only kurtosis cannot distinguish any zero-kurtosis distribution from a Gaussian.

Another measure of non-Gaussianity is information theoretic, relying on the result that: out of all real distributions with equal variance, that with the largest entropy is the Gaussian. One can then define the non-negative *negentropy* of a random variable $\mathbf{x}$

$$J(\mathbf{x}) = H(\mathbf{x}_{\text{Gauss}}) - H(\mathbf{x}) \tag{1.27}$$

where $H(\mathbf{x})$ is the entropy of $\mathbf{x}$, and $\mathbf{x}_{\text{Gauss}}$ is a Gaussian random variable with the same covariance matrix as $\mathbf{x}$. $J(\mathbf{x})$ is zero if and only if $\mathbf{x}$ is Gaussian. Use of negentropy is well justified as a measure of non-Gaussianity, but is difficult to compute, requiring an approximation of the pdf.

A synthetic example is illustrated in **Figure 1.2**. Latent signals are simulated and a mixing matrix is applied. The observed mixed signals are then unmixed using ICA. The latent signals are independent: one dimension (V1) is simulated from a uniform distribution, and the other (V2) from an exponential. The independant signals (a uniform and an exponential distribution) are correctly identified by ICA. Note that ICA is unable to recover the order of the signals, nor their sign.

Sparse PCA

Sparse PCA (SPCA) is closely related to PCA, but produces exact zeroes in the loadings. This sparsity means the loadings no longer capture the maximum variance subspace, but interpretability and computation are aided since only a subset of the

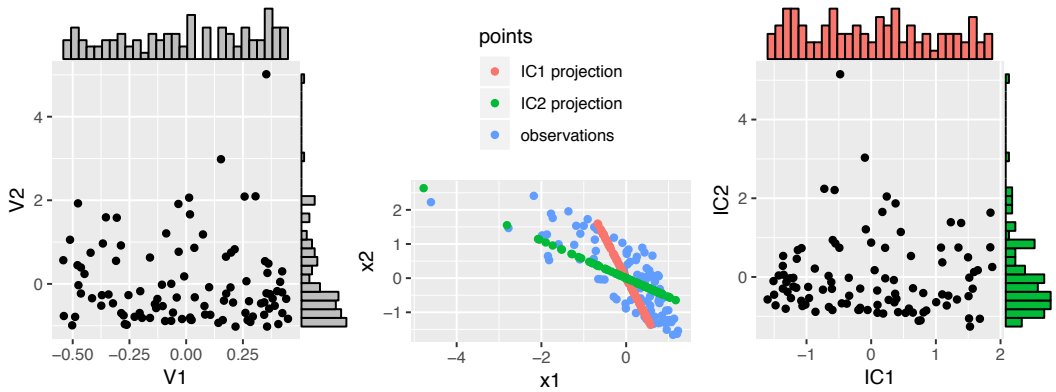


Figure 1.2: *Left: true, latent signals have been generated independently from a uniform (V1) and an exponential (V2) distribution. Centre: Observations, as well as their projections onto the Independent Components. Right: The learned latent space correctly identifies approximately uniform and exponentially distributed latent signals.*

full loadings need to be considered. It is possible for an entire row of the loadings to be zero, thus completely ignoring the associated variable when computing the latent variables. Sparsity is achieved using an ℓ_1 penalty; intuition for why this produces sparsity is given in **Section 5.1.2**.

There are a number of formulations of SPCA. The SCoTLASS procedure for rank-1 SPCA [Jolliffe et al., 2003] solves the minimisation

$$\underset{\substack{||\mathbf{u}||_2=1 \\ ||\mathbf{u}||_1 \leq t}}{\arg\max} \left\{ \mathbf{u}^\top \mathbf{X}^\top \mathbf{X} \mathbf{u} \right\} \tag{1.28}$$

which is the maximum variance formulation of PCA with the additional constraint that the ℓ_1 norm of the vector $\mathbf{u}$ is less than the hyper-parameter t.

Another method of rank-1 SPCA is proposed by Zou et al. [2006], and is based on a reconstruction formulation:

$$\underset{\substack{\mathbf{u},\mathbf{w}\in\mathbb{R}^d \\ ||\mathbf{w}||_2=1}}{\arg\min} \left\{ ||\mathbf{X} - \mathbf{X}\mathbf{u}\mathbf{w}^\top||_{\mathrm{F}}^2 + \lambda_1 ||\mathbf{u}||_1 + \lambda_2 ||\mathbf{u}||_2^2 \right\} \tag{1.29}$$

which is obtained by taking the reconstruction formulation of PCA and applying the elastic net penalty (the ℓ_1 plus ℓ_2 penalty) to the loadings. Typically λ_2 is set to a fixed small constant (e.g., 10^{-6}), whilst λ_1 requires more tuning. PCA is recovered with the settings $\lambda_1 = \lambda_2 = 0$.

One can extend this to a higher rank case by iteratively solving for further sparse PCs constrained to be orthogonal to discovered PCs. However, requiring ex-

act orthogonality may limit the sparsity of solutions, and approximate orthogonality may be preferable [Hastie et al., 2015, Section 8.2.3], in which case one can solve the rank-k version of **Equation 1.29**, where $\mathbf{U}$ and $\mathbf{W}$ are now $d \times k$ matrices, for any $1 \leq k \leq d$:

$$\underset{\substack{\mathbf{U},\mathbf{W}\in\mathbb{R}^{d\times k}\\ \mathbf{W}^\top\mathbf{W}=I_k}}{\arg\min} \left\{ ||\mathbf{X} - \mathbf{X}\mathbf{U}\mathbf{W}^\top||_{\mathrm{F}}^2 + \lambda_1||\mathbf{U}||_1 + \lambda_2||\mathbf{U}||_2^2 \right\} \tag{1.30}$$

Dictionary Learning

Dictionary Learning (DL) was originally developed as a model of the mammalian visual cortex, and was termed Sparse Coding [Olshausen and Field, 1997], though this term is often used to refer to inference of only $\mathbf{V}$ given a fixed $\mathbf{W}$. More efficient algorithms have since been developed [Mairal et al., 2009]. In DL, the learned latent variables $\mathbf{V}$ (also known as the "sparse codes") are typically sparse due to a prior over each latent variable which is highly peaked around zero. This is a similar problem to SPCA except the goal here is to find sparse $\mathbf{V}$ as opposed to sparse $\mathbf{W}$.

Sparsity in $\mathbf{V}$ allows an *overcomplete* dictionary to be learned, where $k > d$. The intuition behind this is that only the dictionary elements needed to explain an observation are utilised by the sparse code. More technically, $p(\mathbf{x}|\mathbf{v}, \mathbf{W})$, as a function of $\mathbf{v}$, will have a non-unique maximum along a ridge. The action of the prior means that $p(\mathbf{x}|\mathbf{v}, \mathbf{W})p(\mathbf{v})$ has a unique maximum (up to permutation and sign) even as $k > d$, where some elements of $\mathbf{V}$ are exactly zero.

Mairal et al. [2009] use the definition of Dictionary Learning as performing the minimisation

$$\begin{aligned} \underset{\substack{\mathbf{V}\in\mathbb{R}^{n\times k}\\ \mathbf{W}\in\mathbb{R}^{d\times k}}}{\arg\min} \quad & \left\{ \frac{1}{2}||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\mathrm{F}}^2 + \lambda||\mathbf{V}||_1 \right\} \\ \text{subject to} \quad & |\mathbf{w}_i| \leq 1 \;\; \forall i = 1, \cdots, k \end{aligned} \tag{1.31}$$

By minimising this loss function, DL trades off ℓ_2 reconstruction error against sparsity in $\mathbf{V}$, where the strength of the trade-off is controlled by the hyper-parameter λ. The constraint on the magnitude of each dictionary element exists because multiplying $\mathbf{W}$ and $\mathbf{V}$ by arbitrary c and c^{-1} respectively leaves the reconstructions unchanged but reduces the penalty $||\mathbf{V}||_1$ for $c > 1$.

Equation 1.31 is non-convex in $\mathbf{V}$ and $\mathbf{W}$ jointly, but convex in each term individually. Alternating between $\mathbf{V}$ and $\mathbf{W}$ thus produces tractable inference techniques. With fixed $\mathbf{V}$, finding the optimum $\mathbf{W}$ is a simple regression with a convex constraints $|\mathbf{w}_i| \leq 1$. With fixed $\mathbf{W}$, the optimum $\mathbf{V}$ is found using a lasso regres-

sion, for which many algorithms exist [e.g., Hastie et al., 2015].

Non-negative Matrix Factorisation

Non-negative Matrix Factorisation (NMF) [Paatero and Tapper, 1994] can be obtained by optimising a constrained ℓ_2 reconstruction of the data:

$$\underset{\substack{\mathbf{W}\in\mathbb{R}^{d\times k}\\ \mathbf{V}\in\mathbb{R}^{n\times k}}}{\arg\min}\ \frac{1}{2}||\mathbf{X}-\mathbf{V}\mathbf{W}^\top||_{\mathrm{F}}^2 \text{ subject to } \mathbf{W}\geq 0,\ \mathbf{V}\geq 0 \tag{1.32}$$

The reconstructions $\mathbf{V}\mathbf{W}^\top$ contain non-negative values. Such an algorithm would be useful when we have prior knowledge of the observations being non-negative (or that it would be non-negative if it were noiseless), meaning the data must *not* be centred. This model is symmetric in features and observations: switching $\mathbf{X}\rightarrow\mathbf{X}^\top$ simply swaps the learned parameters $\mathbf{V},\mathbf{W}$. Since the objective function is unchanged by scaling $\mathbf{V}$ by a constant and $\mathbf{W}$ by the inverse, it may be desirable to introduce constraints or regularisation terms. Note that this model has no hyper-parameters.

Robust PCA

Robust PCA (RPCA) [Candès et al., 2011] is a modification to PCA which enables principal subspace recovery under corrupted observations. The matrix of observations $\mathbf{X}$ is modelled as

$$\mathbf{X}=\mathbf{L}+\mathbf{S} \tag{1.33}$$

where $\mathbf{L}$ is a low-rank matrix, and $\mathbf{S}$ is a sparse matrix. Performing RPCA involves decomposing a dataset $\mathbf{X}$ into $\mathbf{L}$ and $\mathbf{S}$. $\mathbf{L}$ then contains a low-rank approximation to an un-corrupted $\mathbf{X}$, and $\mathbf{S}$ contains the corruptions. The term 'robust' refers to robustness of the technique to outliers. PCA minimises the sum of the squared reconstruction errors, so a single outlying value can have a large impact on the recovered subspace. In RPCA, such an outlier would be absorbed into $\mathbf{S}$, leaving $\mathbf{L}$ to capture the principal subspace of the un-corrupted data.

A number of algorithms exist to produce the decomposition in RPCA. Principal Component Pursuit [Candès et al., 2011] solves:

$$\begin{aligned}\text{minimise}\quad & ||\mathbf{L}||_*+\lambda||\mathbf{S}||_1\\ \text{subject to}\quad & \mathbf{L}+\mathbf{S}=\mathbf{X}\end{aligned} \tag{1.34}$$

$||\mathbf{L}||_*$ is the *nuclear norm* of $\mathbf{L}$, which is the sum of the singular values of $\mathbf{L}$. The hyper-parameter λ, controls the trade-off between producing low-rank $\mathbf{L}$ and sparse

Figure 1.3: *Two frames of a train station surveillance video which have been decomposed into foreground and background using RPCA. Left column: original video frames (rows of* $\mathbf{X}$*). Centre column: foreground (rows of* $\mathbf{L}$*). Right column: background (rows of* $\mathbf{S}$*). Note that* $\mathbf{S}$ *does not really contain the foreground as we would intuitively mean, but contains the perturbations to the background which obtain the foreground.*

$\mathbf{S}$. Performing this minimisation produces sparsity in the singular values of $\mathbf{L}$ and the elements of $\mathbf{S}$. Intuition as to why this produces sparsity is given in **Section 5.1.2**, but the key point is that an ℓ_1 regularised minimisation often produces sparse coefficients, and the nuclear norm can be seen as an ℓ_1 penalty on the singular values of $\mathbf{L}$.

The constraint of $\mathbf{L} + \mathbf{S}$ exactly reconstructing $\mathbf{X}$ may be relaxed [Hastie et al., 2015, Section 7.7]. This is equivalent to augmenting the decomposition as $\mathbf{X} = \mathbf{L} + \mathbf{S} + \mathbf{N}$ where $\mathbf{N}$ contains low-magnitude Gaussian noise).

One interesting application of RPCA is foreground/background separation in video. By considering each of the video frames as an i.i.d row in $\mathbf{X}$ and performing RPCA, the video is separated into the approximately static background in $\mathbf{L}$, and the sparse perturbations in $\mathbf{S}$ which give the foreground. This is illustrated in **Figure 1.3**, which was generated using the code published by Lin et al. [2010].

Gaussian Process Latent Variable Model

The Gaussian Process Latent Variable Model (GP-LVM) [Lawrence, 2004] is a method for nonlinear dimension reduction. The form of the non-linearity is dictated by a choice of covariance function, and a linear covariance function recovers PPCA. The model is dual to PPCA: in the GP-LVM the loadings are integrated

out and the latent variables optimised, in PPCA this is reversed. This leads to a Gaussian process mapping from the latent space to the data space.

As in PPCA, the GP-LVM uses

$$p(\mathbf{x}|\mathbf{v}, \mathbf{W}, \beta) = \mathcal{N}(\mathbf{x}|\mathbf{W}\mathbf{v}, \beta^{-1}I) \tag{1.35}$$

To integrate out the loadings, an independent prior over each element of $\mathbf{W}$ is introduced with hyper-parameter α.

$$p(W_{ij}|\alpha) = \mathcal{N}(W_{ij}|0, \alpha^{-1}) \tag{1.36}$$

This enables a closed form marginalisation over $\mathbf{W}$, producing a marginalised likelihood

$$p(\mathbf{X}|\mathbf{V}, \alpha, \beta) = \int p(\mathbf{X}|\mathbf{V}, \mathbf{W}, \beta)p(\mathbf{W}|\alpha)\,\mathrm{d}\mathbf{W} \tag{1.37}$$

$$= (2\pi)^{-\frac{nd}{2}}|\mathbf{K}|^{-\frac{d}{2}}\exp\left(-\frac{1}{2}\operatorname{Tr}\left[\mathbf{X}^\top\mathbf{K}^{-1}\mathbf{X}\right]\right) \tag{1.38}$$

$$= \prod_{i=1}^{d}\mathcal{N}(\mathbf{X}_{:,i}|\mathbf{0}, \mathbf{K}) \tag{1.39}$$

where $\mathbf{K} = \alpha\mathbf{V}\mathbf{V}^\top + \beta^{-1}I$. This is a product of d independent Gaussian processes mapping from the latent space to each observed variable. The covariance function for the GP mapping is $K_{ij} = \alpha\mathbf{v}_i^\top\mathbf{v}_j + \beta^{-1}\mathbb{1}[i = j]$, which is a linear plus a noise covariance function. One can then consider other covariance functions allowing non-linear mappings.

Regardless of covariance function, one can optimise $\mathbf{V}$ with a non-linear gradient based optimiser. Gradients of the log likelihood can be obtained via the chain rule using

$$\frac{\partial \ln p(\mathbf{X}|\mathbf{V}, \alpha, \beta)}{\partial \mathbf{K}} = \mathbf{K}^{-1}\mathbf{V}\mathbf{V}^\top\mathbf{K}^{-1} - d\mathbf{K}^{-1} \tag{1.40}$$

along with $\frac{\partial \mathbf{K}}{\partial v_{ij}}$, which will depend on the choice of covariance function. Similarly, kernel hyper-parameters such as β may be tuned jointly with $\mathbf{V}$.

Autoencoders

An autoencoder is a neural network trained to output its own input. If the network is constrained in some way, such as having a hidden layer of dimensionality smaller than the input, then the network is unable to learn the identity function. Thus, to

produce an output similar to the input, the network learns to represent the data in a way such that accurate reconstructions can be made for inputs which are likely, foregoing the ability to reconstruct unlikely inputs. The actual reconstruction is usually not of interest, but the latent representation can be used like that of any other model discussed. Goodfellow et al. [2016, Chapter 14] provides an excellent overview of the area.

An autoencoder consists of an encoder function $\mathbf{v} = f(\mathbf{x})$ and a decoder function $\mathbf{x}_{\text{rec}} = g(\mathbf{v})$. These are both parametrised, and are trained to minimise a loss function $\mathcal{L}(\mathbf{x}, g(f(\mathbf{x})))$ which penalises dissimilarity between $\mathbf{x}$ and its reconstruction $\mathbf{x}_{\text{rec}}$. $\mathbf{v}$ is a latent representation of $\mathbf{x}$, and is the object of interest for feature learning. Being a neural network, training is usually performed with some variant of stochastic gradient descent [Goodfellow et al., 2016, Chapter 8].

Many of the above models can be formulated as autoencoders, with PCA being a simple example [Baldi and Hornik, 1989]. If we have a linear encoder and decoder

$$\mathbf{v} = \mathbf{W}\mathbf{x} \tag{1.41}$$

$$\mathbf{x}_{\text{rec}} = \mathbf{U}\mathbf{v} \tag{1.42}$$

and a squared error loss function:

$$\mathcal{L}(\mathbf{x}, g(f(\mathbf{x}))) = ||\mathbf{x} - \mathbf{x}_{\text{rec}}||_2^2 = ||\mathbf{x} - \mathbf{U}\mathbf{W}\mathbf{x}||_2^2 \tag{1.43}$$

then minimising the loss

$$\sum_{i=1}^{n} \mathcal{L}(\mathbf{x}_i, g(f(\mathbf{x}_i))) \tag{1.44}$$

reduces to performing the minimisation

$$\underset{\mathbf{W},\mathbf{U}}{\arg\min}\ ||\mathbf{X} - \mathbf{X}\mathbf{W}^\top\mathbf{U}^\top||_{\text{F}}^2 \tag{1.45}$$

The minimum loss in **Equation 1.45** is achieved when $\mathbf{W}^\top\mathbf{U}^\top$ is the orthogonal projection onto the principal subspace of $\mathbf{X}$. In this case, $\mathbf{W}$ spans the principal subspace, and the left singular vectors of $\mathbf{W}$, given by SVD, are the PCs.

SPCA or Sparse Coding can be reached from such an autoencoder (single hidden layer, identity activation function) by adding a lasso penalty to the loss function over $\mathbf{W}$ or $\mathbf{V}$ respectively. One could also partition $\mathbf{V}$ into equal sized $\mathbf{V}_{\text{L}}$, $\mathbf{V}_{\text{S}}$ and change the decoder to $\mathbf{X}_{\text{rec}} = (\mathbf{V}_{\text{L}} + \mathbf{V}_{\text{S}})\mathbf{U}$. By then adding an ℓ_1 norm penalty to $\mathbf{V}_{\text{S}}$ and a nuclear norm penalty to $\mathbf{V}_{\text{L}}$, Robust PCA is recovered.

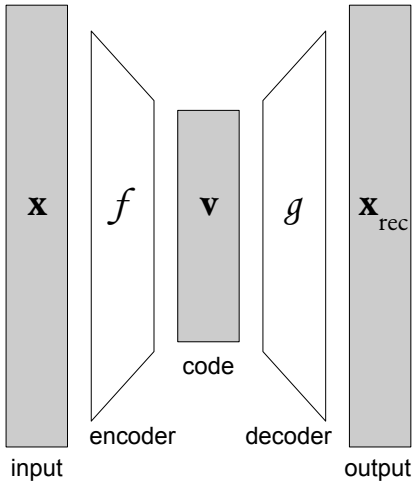


Figure 1.4: *Graphical representation of an autoencoder. The input vector is fed through the encoder which outputs a (typically lower dimension) code, which when input into the decoder gives an imperfect reconstruction of the input.*

Formulating these models as autoencoders is enlightening and highlights how they are linked. However, the methods of inference typically used in autoencoders (e.g., stochastic gradient descent) would perform poorly compared to model-specific inference techniques. Autoencoders in practice use a deep neural network for both the encoder and decoder with non-linear activation functions, allowing for highly non-linear feature extractors to be learned.

Summary

A number of latent variable models have been presented, and their salient features are compared in **Table 1.1**. The models presented were each chosen to illustrate a single way in which PCA can be extended, but it is possible to combine multiple of these concepts. Mairal et al. [2010] allow sparsity constraints on both the loadings and the latent variables leading to a method with sparsity in both $\mathbf{W}$ and $\mathbf{V}$. Hubert et al. [2016] detail a sparse and robust PCA.

1.1.2 Inference

Expectation Maximisation

Expectation Maximisation (EM) is an algorithm for finding the maximum likelihood or maximum-a-posteriori parameters in a latent variable model. EM is typically used when directly maximising the likelihood is difficult, but maximising the *joint likelihood* $p(\mathbf{X}, \mathbf{V}|\theta)$ is easier. This is the case for Factor Analysis where $\arg\max_\theta\ p(\mathbf{X}|\theta)$

PCA	Linear mapping justified as maximum variance preserving, or minimum reconstruction error.
PPCA	Probabilistic model: MAP parameters recover principal subspace.
FA	Generalises PPCA with a separate parameter for noise on each observed variable.
Categorical PCA	Categorical likelihood considers discrete-valued data.
ICA	Learn loadings such that latent variables are independent.
SPCA	PCA variant with sparsity in loadings.
DL	PCA variant with sparsity in latent variables.
NMF	Loadings and latent variables both non-negative.
RPCA	Decompose matrix of data into sparse + low rank matrices.
GP-LVM	Nonlinear; data are modelled as Gaussian Process mapping from latent space.
Autoencoder	Generalises many of the above. Usually deep (highly non-linear) architecture.

Table 1.1: *List of latent variable models presented in this section, as well as their most salient feature when compared to classical PCA.*

does not have a solution in closed form, but $\arg\max_\theta\, p(\mathbf{X}, \mathbf{V}|\theta)$ does.

EM describes an alternating scheme with an expectation step (*E-step*) and a maximisation step (*M-step*), beginning with some guess at the parameters θ^{old}. In the E-step, we compute the posterior over latent variables $p(\mathbf{V}|\mathbf{X}, \theta^{\text{old}})$, and construct a function giving the expected complete log likelihood $\mathbb{E}[\log p(\mathbf{X}, \mathbf{V}|\theta)]_{p(\mathbf{V}|\mathbf{X},\theta^{\text{old}})}$. This often simplifies, especially in the case where $p(\mathbf{X}, \mathbf{V}|\theta)$ is in the exponential family, since the log cancels the exponentiation. In the case of a Gaussian complete log likelihood, this reduces to computing moments, which we will see. In the M-step, the expected complete log likelihood is maximised with respect to θ. This EM cycle is repeated until convergence.

This procedure converges to the maximum likelihood parameters, and we walk through a proof of this below. The proof is well known, and involves decomposing $\log p(\mathbf{X}|\theta)$ into a lower bound plus a non-negative residual. The E-step and M-step are shown to be a consequence of maximising this lower bound.

We first introduce $q(\mathbf{V})$ which is any arbitrary distribution over the latent variables with the same support as $p(\mathbf{V}|\mathbf{X}, \theta)$. Since $q(\mathbf{V})$ integrates to 1, it is trivially true that

$$\log p(\mathbf{X}|\theta) = \int q(\mathbf{V}) \log p(\mathbf{X}|\theta)\, \mathrm{d}\mathbf{V} \tag{1.46}$$

By the product rule, $p(\mathbf{X}|\theta) = p(\mathbf{X}, \mathbf{V}|\theta)/p(\mathbf{V}|\mathbf{X}, \theta)$. We use this below, introduce

$q(\mathbf{V})/q(\mathbf{V})$ and split the equation into two terms.

$$\log p(\mathbf{X}|\theta) = \int q(\mathbf{V}) \log \frac{p(\mathbf{X}, \mathbf{V}|\theta)}{p(\mathbf{V}|\mathbf{X}, \theta)} \, \mathrm{d}\mathbf{V} \tag{1.47}$$

$$= \int q(\mathbf{V}) \log \frac{q(\mathbf{V}) p(\mathbf{X}, \mathbf{V}|\theta)}{q(\mathbf{V}) p(\mathbf{V}|\mathbf{X}, \theta)} \, \mathrm{d}\mathbf{V} \tag{1.48}$$

$$\begin{aligned} &= \int q(\mathbf{V}) \log \frac{p(\mathbf{X}, \mathbf{V}|\theta)}{q(\mathbf{V})} \, \mathrm{d}\mathbf{V} \\ &\quad - \int q(\mathbf{V}) \log \frac{p(\mathbf{V}|\mathbf{X}, \theta)}{q(\mathbf{V})} \, \mathrm{d}\mathbf{V} \end{aligned} \tag{1.49}$$

$$= \mathcal{L}(q, \theta) + \mathrm{KL}(q||p) \tag{1.50}$$

$\mathrm{KL}(q||p)$ is the Kullback-Leibler divergence of the distribution q from the distribution $p(\mathbf{V}|\mathbf{X}, \theta)$. This is a non-negative quantity, with $\mathrm{KL}(q||p) = 0$ holding if and only if $q(\mathbf{V}) = p(\mathbf{V}|\mathbf{X}, \theta)$ everywhere. Since $\mathrm{KL}(q||p)$ is non-negative, $\mathcal{L}(q, \theta)$ is a lower bound on $\log p(\mathbf{X}|\theta)$. We can thus maximise $\log p(\mathbf{X}|\theta)$ by maximising $\mathcal{L}(q, \theta)$, as long as $\mathrm{KL}(q||p) = 0$ at this maximum, which we will see to be true. Iterating between maximisation with respect to q and θ turns out to correspond to the E-step and M-step respectively.

If we have some current value of the parameters θ_{old}, we can maximise $\mathcal{L}(q, \theta)$ with respect to q by considering that this maximum must occur when $\mathrm{KL}(q||p) = 0$, meaning $q = p$. This gives us the solution that $q(\mathbf{V}) = p(\mathbf{V}|\mathbf{X}, \theta_{\mathrm{old}})$. By substituting this into the lower bound we arrive at

$$\mathcal{L}(p, \theta) = \int p(\mathbf{V}|\mathbf{X}, \theta_{\mathrm{old}}) \log \frac{p(\mathbf{X}, \mathbf{V}|\theta)}{p(\mathbf{V}|\mathbf{X}, \theta_{\mathrm{old}})} \, \mathrm{d}\mathbf{V} \tag{1.51}$$

$$= \int p(\mathbf{V}|\mathbf{X}, \theta_{\mathrm{old}}) \log p(\mathbf{X}, \mathbf{V}|\theta) \, \mathrm{d}\mathbf{V} + \mathrm{const} \tag{1.52}$$

$$= \mathbb{E}\left[\log p(\mathbf{X}, \mathbf{V}|\theta)\right]_{p(\mathbf{V}|\mathbf{X}, \theta_{\mathrm{old}})} + \mathrm{const} \tag{1.53}$$

$$=: Q(\theta, \theta_{\mathrm{old}}) + \mathrm{const} \tag{1.54}$$

$$\tag{1.55}$$

This is the E-step of EM. The expectation can be computed in closed form in a number of models, and frequently involves computing a set of sufficient statistics. Note that $\mathcal{L}(p, \theta_{\mathrm{old}})$ is equal to $\log p(\mathbf{X}|\theta_{\mathrm{old}})$ since the KL term is zero.

We have maximised $\mathcal{L}(q, \theta)$ with respect to q, and we can now maximise with respect to θ. For any θ such that $Q(\theta, \theta_{\mathrm{old}}) > Q(\theta_{\mathrm{old}}, \theta_{\mathrm{old}})$, it will hold that $p(\mathbf{X}|\theta) > p(\mathbf{X}|\theta_{\mathrm{old}})$. However, this will also cause the (typically unknown) KL term to become positive, so a $\mathcal{L}(q, \theta)$ will again become a lower bound.

Maximising $\mathcal{L}(q, \theta)$ with respect to θ is the M-step in EM. It is sufficient to maximise $Q(\theta, \theta_{\text{old}})$, which is the expected complete-data log likelihood. In many models, including PPCA and FA, $p(\mathbf{X}|\mathbf{V}, \theta)$ and $p(\mathbf{V})$ both belong to the exponential family, producing a closed-form maximisation. Whilst EM specifies full maximisation in the M-step, any increase in the lower bound during the M-step will also increase the likelihood and thus converge to a maxima, and is known as Generalised EM.

Example: Expectation Maximisation for PPCA

EM is useful if maximising $Q(\theta, \theta_{\text{old}})$ is easier than maximising $p(\mathbf{X}|\theta)$. This is often the case for the exponential family, where the log cancels the exponentiation. EM for PPCA illustrates this; the E-step is computed as:

$$\begin{aligned}\mathbb{E}\left[\log p(\mathbf{X}, \mathbf{V}|\theta)\right]_{p(\mathbf{V}|\mathbf{X}, \theta_{\text{old}})} \propto -\frac{1}{2}\sum_{i=1}^{n}\Big\{ & d\log\sigma^2 + \sigma^{-2}||\mathbf{x}_i||^2 \\ & - 2\sigma^{-2}\mathbb{E}[\mathbf{v}_i]^\top \mathbf{W}^\top \mathbf{x}_i \\ & + \sigma^{-2}\,\text{Tr}\Big[\mathbb{E}[\mathbf{v}_i\mathbf{v}_i^\top]\mathbf{W}^\top\mathbf{W}\Big] \\ & + \text{Tr}\Big[\mathbb{E}[\mathbf{v}_i\mathbf{v}_i^\top]\Big]\Big\}\end{aligned} \tag{1.56}$$

The full expectation has reduced to computing just the sufficient statistics of the Gaussian. This requires $p(\mathbf{v}_i|\mathbf{x}_i, \theta)$ (**Equation 1.20**) and uses the identity $\mathbb{E}[\mathbf{v}_i\mathbf{v}_i^\top] = \text{cov}[\mathbf{v}_i] + \mathbb{E}[\mathbf{v}_i]\mathbb{E}[\mathbf{v}_i]^\top$, giving:

$$\mathbb{E}[\mathbf{v}_i] = \mathbf{M}_{\text{old}}^{-1}\mathbf{W}_{\text{old}}^\top\mathbf{x}_i \tag{1.57}$$

$$\mathbb{E}[\mathbf{v}_i\mathbf{v}_i^\top] = \sigma_{\text{old}}^2(\mathbf{W}_{\text{old}}^\top\mathbf{W}_{\text{old}} + \sigma_{\text{old}}^2 I)^{-1} + \mathbb{E}[\mathbf{v}_i]\mathbb{E}[\mathbf{v}_i]^\top \tag{1.58}$$

To find θ_{new}, the M-step may then be performed by taking derivatives of **Equation 1.56** with respect to θ, setting to zero and solving analytically. This gives the update step:

$$\mathbf{W}_{\text{new}} = \left[\sum_{i=1}^{n}\mathbf{x}_i\mathbb{E}[\mathbf{v}_i]^\top\right]\left[\sum_{i=1}^{n}\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]\right]^{-1} \tag{1.59}$$

$$\begin{aligned}\sigma_{\text{new}}^2 = & \frac{1}{nd}\sum_{i=1}^{n}\Big\{\mathbf{x}_i\mathbf{x}_i^\top - 2\mathbb{E}[\mathbf{v}_i]^\top\mathbf{W}_{\text{new}}^\top\mathbf{x}_i \\ & + \text{Tr}\Big[\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]\mathbf{W}_{\text{new}}^\top\mathbf{W}_{\text{new}}\Big]\Big\}\end{aligned} \tag{1.60}$$

The fact that $\mathbf{W}_{\text{new}}$ does not depend on σ^2_{new} means that calculating $\mathbf{W}_{\text{new}}$ and then σ^2_{new} gives us the joint maxima of $\mathbb{E}\big[\log p(\mathbf{X}, \mathbf{V}|\mathbf{W}, \sigma^2)\big]$.

Gradient Ascent

One of the simplest techniques to find the maximum likelihood parameters is gradient ascent [see e.g., Goodfellow et al., 2016, Section 4.3]. The idea is that one can think of the log likelihood as a high dimensional surface on which we try and find the highest point by repeatedly taking small steps in the direction pointing most steeply uphill. This requires some initial value of the parameters $\theta^{(0)}$ and the ability to compute the gradient $\frac{\partial \log p(\mathbf{X}|\theta)}{\partial \theta}$. Taking a step in the uphill direction corresponds to computing the gradient ascent step:

$$\theta^{(t+1)} = \theta^{(t)} + \eta \frac{\partial \log p(\mathbf{X}|\theta)}{\partial \theta}\Big|_{\theta=\theta^{(t)}} \tag{1.61}$$

This is iterated until some convergence criteria is met, such as the gradient becoming sufficiently small. The learning rate η is a hyper-parameter and controls the size of the step taken. If η is too large then the gradient steps may decrease the log likelihood, resulting in poor performance. Using a small value of η will reduce the risk of this, at the cost of slower convergence.

Gradient ascent can fail in very flat regions of the likelihood landscape. If $\theta^{(0)}$ is chosen poorly, such as a value with extremely low likelihood, the gradient may not point towards the maxima, or may become numerically indistinguishable from zero. Saddle points may also be problematic; since the gradient nearby will be very close to zero, very small steps may be taken leading to slow or premature convergence.

Basic gradient ascent can be modified in a number of ways. Instead of fixing η, each gradient step may be evaluated with multiple values of η and the step is made with the value causing the biggest increase in the log likelihood. This is known as *line search*.

Gradient ascent may perform poorly if the log likelihood changes rapidly in some directions and slowly in others, producing a surface looking like a long ridge. Ideally each gradient step would move more in the direction of low variance. If we can compute the Hessian, once can instead use the step:

$$\theta^{(t+1)} = \theta^{(t)} + \mathbf{H}^{-1}_{\theta^{(t)}} \frac{\partial \log p(\mathbf{X}|\theta)}{\partial \theta}\Big|_{\theta=\theta^{(t)}} \tag{1.62}$$

which uses the Hessian

$$\mathbf{H}_{\theta^{(t)}} = \frac{\partial^2 \log p(\mathbf{X}|\theta)}{\partial\theta\partial\theta^\top}\Big|_{\theta=\theta^{(t)}} \tag{1.63}$$

This is *Newton's method* [see e.g., Goodfellow et al., 2016, Section 4.3.1], which consists of fitting a quadratic to the function being minimised, and jumping straight to the maxima. This can produce faster convergence as long as the log likelihood is locally concave, however the Hessian may be very large if there are many parameters.

1.2 Statistical Background

1.2.1 Bayesian Model Selection

A common scenario in a scientific analysis is to have a collection of models $\mathcal{M}_1, \mathcal{M}_2, \cdots$ describing the distribution of some observations of interest of through $p(\mathbf{x}|\mathcal{M})$. Given this set, we would like to select a single model for use in an analysis. To do this we require a definition of which is the "best" model out of a collection. In Bayesian Model Selection, we first observe a set of data $\mathbf{X}$ and then select the most likely model given that data, which is the model with the greatest posterior probability $p(\mathcal{M}|\mathbf{X})$.

In a fully Bayesian context, one would not select a single model but would integrate over all models under consideration. However, we may prefer a single model due to interpretability and computation. Assuming identifiability, as the amount of data we observe increases, the mass of $p(\mathcal{M}|\mathbf{X})$ will typically concentrate around a single model. This justifies the use of the *maximum-a-posteriori* model as an approximation to a mixture of every model.

If we have a uniform prior over our models, then our posterior over models is proportional to our likelihood over models:

$$p(\mathcal{M}|\mathbf{X}) = \frac{p(\mathbf{X}|\mathcal{M})p(\mathcal{M})}{\int p(\mathbf{X}|\mathcal{M})p(\mathcal{M})\,\mathrm{d}\mathcal{M}} \tag{1.64}$$

$$= p(\mathbf{X}|\mathcal{M})\frac{p(\mathcal{M})}{p(\mathbf{X})} \tag{1.65}$$

$$\propto p(\mathbf{X}|\mathcal{M}) \tag{1.66}$$

With $p(\mathcal{M}|\mathbf{X}) \propto p(\mathbf{X}|\mathcal{M})$, to select the most likely model it is sufficient to

find the model maximising the quantity

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M}) \, \mathrm{d}\boldsymbol{\theta} \tag{1.67}$$

known as the evidence for model $\mathcal{M}$. This term also appears as the normalisation constant in the posterior over the vector of model parameters $\boldsymbol{\theta}$:

$$p(\boldsymbol{\theta}|\mathbf{X}, \mathcal{M}) = \frac{p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M})}{p(\mathbf{X}|\mathcal{M})} \tag{1.68}$$

The integral in **Equation 1.67** can be computed in closed form if we have an appropriate likelihood with a conjugate prior [e.g., Bishop, 2006, Section 2.4.2], but this is not true in general. The Laplace approximation may be used to approximate the model evidence, which is discussed below

Bayesian Occam's razor

Selecting the model with the greatest evidence performs a trade-off between selecting a model complex enough to fit the data well, but simple enough so that it cannot explain any possible set of observations. We can see this by noting the evidence $p(\mathbf{x}|\mathcal{M})$ defines a normalised distribution over datasets. If model $\mathcal{M}_{\text{complex}}$ is very complex, a wide range of datasets will be considered likely. If model $\mathcal{M}_{\text{simple}}$ is simple, only a small number of datasets will be considered likely. Since these distributions are normalised, the most likely dataset under $\mathcal{M}_{\text{simple}}$ will produce a greater evidence than the most likely dataset under $\mathcal{M}_{\text{complex}}$. Thus, if $\mathcal{M}_{\text{simple}}$ explains the data well, it will be preferred under the Bayesian paradigm. Bayesian inference thus embodies an implicit Occam's razor, preferring explanations that are only complex enough to explain the data observed [e.g., MacKay, 2002, Chapter 28].

Bayes' Factors

It may be desirable to quantify by how much a given model is preferred over another. The model with the highest *maximum-a-posteriori* value may be very complex, and a model which is only slightly inferior (in terms of posterior probability) may be more interpretable or computationally lighter. If we are to compare two models $\mathcal{M}_1$ and $\mathcal{M}_2$, we can compute the Bayes Factor

$$K = \frac{p(\mathcal{M}_1|\mathbf{X})}{p(\mathcal{M}_2|\mathbf{X})} = \frac{p(\mathbf{X}|\mathcal{M}_1) p(\mathcal{M}_1)}{p(\mathbf{X}|\mathcal{M}_2) p(\mathcal{M}_2)} \tag{1.69}$$

$\log_{10} K$	Strength of evidence
< 0	negative (supports $\mathcal{M}_2$)
0 to 0.5	barely worth mentioning
0.5 to 1	substantial
1 to 1.5	strong
1.5 to 2	very strong
2 to 2.5	decisive

Table 1.2: *Interpretation of the scale of a given Bayes' factor* $\frac{p(\mathcal{M}_1|\mathbf{X})}{p(\mathcal{M}_2|\mathbf{X})}$ *in terms of the support for model* $\mathcal{M}_1$ *over* $\mathcal{M}_2$ *[Jeffreys, 1998, Appendix B: Tables of K].*

where we have applied Bayes' rule and the $p(\mathbf{X})$ terms have cancelled. With a uniform prior over models this reduces to the ratio of model evidences. The Bayes' factor can be interpreted as the amount of support for model $\mathcal{M}_1$ over $\mathcal{M}_2$. A number of interpretations have been proposed for different values of K [Jeffreys, 1998; Kass and Raftery, 1995], and we list those given by Jeffreys [1998] in **Table 1.2**.

Bayes' Factors also have the desirable property of being applicable even under improper priors, as long as the parameters with the improper priors are shared between the models being compared. Improper priors lead to improper evidence, which can lead to difficulties when comparing model evidences since each evidence is multiplied by an unknown constant, and this constant may differ between models. However, when the only improper priors are shared between models then this unknown constant is the same for each model and thus evidence comparison is possible [Robert, 2007, Chapter 7].

The Laplace Approximation

Suppose that we are interested in a probability distribution $q(\mathbf{x}) = \frac{\tilde{q}(\mathbf{x})}{Z_q}$, where we know the un-normalised distribution $\tilde{q}(\mathbf{x})$ but do not know the intractable normalising constant $Z_q = \int \tilde{q}(\mathbf{x})\, \mathrm{d}\mathbf{x}$. We may approximate $q(\mathbf{x})$ with a Gaussian distribution centred at $\hat{\mathbf{x}} = \arg\max_{\mathbf{x}} \tilde{q}(\mathbf{x})$, which we can find analytically or numerically.

The Laplace approximation of $q(\mathbf{x})$ gives us an approximating Gaussian distribution, and can be justified in terms of a truncated Taylor expansion. If we Taylor expand $\log \tilde{q}(\mathbf{x})$ around $\hat{\mathbf{x}}$ and drop terms above second order, we get an un-normalised Gaussian approximation to $\tilde{q}(\mathbf{x})$.

$$\log \tilde{q}(\mathbf{x}) = \log \tilde{q}(\hat{\mathbf{x}}) + \nabla_{\mathbf{x}} \log \tilde{q}\big|_{\hat{\mathbf{x}}}^{\top} (\mathbf{x} - \hat{\mathbf{x}}) - \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^{\top} \mathbf{H} (\mathbf{x} - \hat{\mathbf{x}}) + \dots \tag{1.70}$$

$$\approx \log \tilde{q}(\hat{\mathbf{x}}) - \frac{1}{2}(\mathbf{x} - \hat{\mathbf{x}})^{\top} \mathbf{H} (\mathbf{x} - \hat{\mathbf{x}}) \tag{1.71}$$

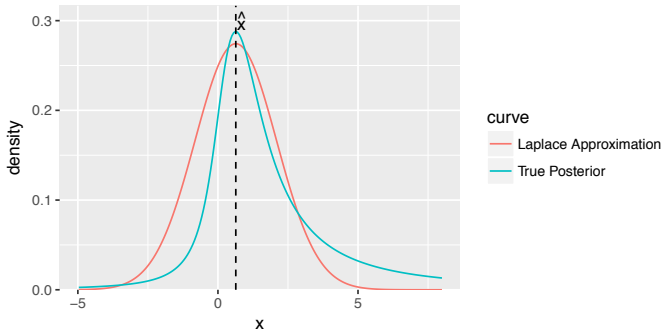


Figure 1.5: *Laplace approximation (red) to a non-Gaussian distribution*

Where $\mathbf{H}_{ij} = -\frac{\partial^2 \log \tilde{q}}{\partial x_i \partial x_j}\big|_{\hat{\mathbf{x}}}$ is the Hessian of $-\log \tilde{q}(\mathbf{x})$ evaluated at $\hat{\mathbf{x}}$. Since $\hat{\mathbf{x}}$ is a maximum of $\tilde{q}(\mathbf{x})$, the gradient $\nabla_{\mathbf{x}} \log \tilde{q}\big|_{\hat{\mathbf{x}}}$ is a vector of zeroes, and $\mathbf{H}$ is nonnegative definite (positive definite if $\tilde{q}(\mathbf{x})$ has a unique maximum, which we assume here).

Exponentiating **Equation 1.71**, we get an un-normalised Gaussian

$$\tilde{q}(\mathbf{x}) \approx \tilde{q}(\hat{\mathbf{x}}) \exp\left\{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathbf{H} (\mathbf{x}-\hat{\mathbf{x}})\right\} \tag{1.72}$$

This may be normalised in closed form to approximate $q(\mathbf{x})$

$$q(\mathbf{x}) \approx \frac{1}{Z_{\text{laplace}}} \tilde{q}(\hat{\mathbf{x}}) \exp\left\{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathbf{H} (\mathbf{x}-\hat{\mathbf{x}})\right\} \tag{1.73}$$

where

$$Z_{\text{laplace}} = \int \tilde{q}(\hat{\mathbf{x}}) \exp\left\{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathbf{H} (\mathbf{x}-\hat{\mathbf{x}})\right\} \mathrm{d}\mathbf{x} \tag{1.74}$$

$$= \tilde{q}(\hat{\mathbf{x}}) (2\pi)^{\frac{|\mathbf{x}|}{2}} |\mathbf{H}|^{-\frac{1}{2}} \tag{1.75}$$

where $|\mathbf{x}|$ is the length of the vector $\mathbf{x}$. We thus obtain the closed form approximation

$$q(\mathbf{x}) \approx (2\pi)^{-\frac{|\mathbf{x}|}{2}} |\mathbf{H}|^{\frac{1}{2}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\hat{\mathbf{x}})^\top \mathbf{H} (\mathbf{x}-\hat{\mathbf{x}})\right\} \tag{1.76}$$

$$= \mathcal{N}\left(\mathbf{x} \,|\, \hat{\mathbf{x}}, \mathbf{H}^{-1}\right) \tag{1.77}$$

Approximating the model evidence

We may use the Laplace approximation to arrive at an approximation for the model evidence. Using **Equation 1.73**, we can choose $\tilde{q}(\boldsymbol{\theta}) = p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M})$ and

approximate the evidence as follows:

$$p(\mathbf{X}|\mathcal{M}) = \int p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) p(\boldsymbol{\theta}|\mathcal{M}) \, \mathrm{d}\boldsymbol{\theta} \tag{1.78}$$

$$= \int \tilde{q}(\boldsymbol{\theta}) \, \mathrm{d}\boldsymbol{\theta} \tag{1.79}$$

$$\approx \tilde{q}(\hat{\boldsymbol{\theta}}) \int \exp\left\{ -\frac{1}{2} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}})^\top \mathbf{H} (\boldsymbol{\theta} - \hat{\boldsymbol{\theta}}) \right\} \, \mathrm{d}\boldsymbol{\theta} \tag{1.80}$$

$$= \tilde{q}(\hat{\boldsymbol{\theta}}) (2\pi)^{\frac{|\boldsymbol{\theta}|}{2}} |\mathbf{H}|^{-\frac{1}{2}} \tag{1.81}$$

$$= p(\mathbf{X}|\hat{\boldsymbol{\theta}}, \mathcal{M}) p(\hat{\boldsymbol{\theta}}|\mathcal{M}) (2\pi)^{\frac{|\boldsymbol{\theta}|}{2}} |\mathbf{H}|^{-\frac{1}{2}} \tag{1.82}$$

Equation 1.82 can give an estimate of the evidence of model $\mathcal{M}$ as long as we can compute $|\mathbf{H}|$. We can numerically approximate $\mathbf{H}$ and calculate the determinant, but this scales poorly for models with high dimension. Without special structure, storing $\mathbf{H}$ requires $\mathcal{O}(|\boldsymbol{\theta}|^2)$ storage. For very high dimensional problems, this may be prohibitively large.

Bayesian Information Criterion

Further approximations can be made to **Equation 1.82** which arrive at the definition of the Bayesian Information Criterion (BIC) [see e.g., Murphy, 2012, Section 5.3.2.4]. These steps involve deriving a computationally cheaper approximation to $|\mathbf{H}|$ term, then dropping constants and terms related to the prior.

First we expand out the definition of $\mathbf{H}$ and use the i.i.d assumption of our data to expand the log likelihood into a sum of the contributions from each datum.

$$\mathbf{H} = -\frac{\partial^2 \log \tilde{q}}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \Big|_{\hat{\boldsymbol{\theta}}} \tag{1.83}$$

$$= -\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \left\{ \log p(\mathbf{X}|\boldsymbol{\theta}, \mathcal{M}) + \log p(\boldsymbol{\theta}|\mathcal{M}) \right\} \Big|_{\hat{\boldsymbol{\theta}}} \tag{1.84}$$

$$= -\frac{\partial^2}{\partial \boldsymbol{\theta} \partial \boldsymbol{\theta}^\top} \left\{ \frac{1}{n} \sum_{i=1}^{n} n \log p(\mathbf{x}_i|\boldsymbol{\theta}, \mathcal{M}) + \log p(\boldsymbol{\theta}|\mathcal{M}) \right\} \Big|_{\hat{\boldsymbol{\theta}}} \tag{1.85}$$

Now since each $\mathbf{x}_i$ is an i.i.d draw from the true data generating distribution, the

sum here approximates an expectation as n becomes large.

$$\mathbf{H} \approx -\frac{\partial^2}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\left\{\mathbb{E}\left[n\log p(\mathbf{x}|\boldsymbol{\theta},\mathcal{M})\right]_{p_{\text{true}}(\mathbf{x})} + \log p(\boldsymbol{\theta}|\mathcal{M})\right\}\Big|_{\hat{\boldsymbol{\theta}}} \tag{1.86}$$

$$= -n\frac{\partial^2\mathbb{E}\left[\log p(\mathbf{x}|\boldsymbol{\theta},\mathcal{M})\right]}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\Big|_{\hat{\boldsymbol{\theta}}} - \frac{\partial^2\log p(\boldsymbol{\theta}|\mathcal{M})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\Big|_{\hat{\boldsymbol{\theta}}} \tag{1.87}$$

$$= n\mathcal{I}(\hat{\boldsymbol{\theta}}) - \frac{\partial^2\log p(\boldsymbol{\theta}|\mathcal{M})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\Big|_{\hat{\boldsymbol{\theta}}} \tag{1.88}$$

where $\mathcal{I}(\hat{\boldsymbol{\theta}})$ is the Fisher information matrix evaluated at $\hat{\boldsymbol{\theta}}$. This is an important tool in frequentist statistics, but is not vital for the discussion presented here. However, a few properties are worth noting. The expectation is taken over a single sample, and thus $\mathcal{I}(\hat{\boldsymbol{\theta}})$ does not depend on the sample size n.

At this stage it is easy to derive the Bayesian Information Criterion (BIC). Plugging **Equation 1.88** into **Equation 1.82** and taking logarithms, we arrive at an approximation to the evidence

$$\log p(\mathbf{X}|\mathcal{M}) \approx \log p(\mathbf{X}|\hat{\boldsymbol{\theta}},\mathcal{M}) + \log p(\hat{\boldsymbol{\theta}}|\mathcal{M}) + \frac{|\boldsymbol{\theta}|}{2}\log(2\pi) - \frac{1}{2}\log\left|n\mathcal{I}(\hat{\boldsymbol{\theta}}) - \frac{\partial^2\log p(\boldsymbol{\theta}|\mathcal{M})}{\partial\boldsymbol{\theta}\partial\boldsymbol{\theta}^\top}\Big|_{\hat{\boldsymbol{\theta}}}\right| \tag{1.89}$$

To arrive at BIC we multiply by -2 and make two further approximations. Firstly, we assume that our prior $p(\boldsymbol{\theta}|\mathcal{M})$ is broad enough to be treated as a constant, allowing us to drop both terms involving the prior.

$$-2\log p(\mathbf{X}|\mathcal{M}) \approx -2\log p(\mathbf{X}|\hat{\boldsymbol{\theta}},\mathcal{M}) - |\boldsymbol{\theta}|\log(2\pi) + \log|n\mathcal{I}(\hat{\boldsymbol{\theta}})| \tag{1.90}$$

$$= -2\log p(\mathbf{X}|\hat{\boldsymbol{\theta}},\mathcal{M}) + |\boldsymbol{\theta}|\left(\log n - \log(2\pi)\right) + \log|\mathcal{I}(\hat{\boldsymbol{\theta}})| \tag{1.91}$$

Now if we assume n is very large, we can drop any terms not depending on n, arriving at the definition of BIC:

$$\text{BIC} = |\boldsymbol{\theta}|\log n - 2\log p(\mathbf{X}|\hat{\boldsymbol{\theta}},\mathcal{M}) \tag{1.92}$$

BIC is widely used for model selection, largely because it is very easy to compute, but is not always appropriate. A more accurate way of performing model selection would be to pick the model maximising **Equation 1.82**, but this may be infeasible if $|\boldsymbol{\theta}|$ is large.

1.2.2 Empirical Bayes

Empirical Bayes [see e.g., Murphy, 2012, Section 5.6] is an approximation to full Bayesian inference in a hierarchical model $p(\mathbf{X},\theta,\beta) = p(\mathbf{X}|\theta)p(\theta|\beta)p(\beta)$. In the

full Bayesian treatment one would often integrate out the hyper-parameters, but this is often computationally prohibitive. The Empirical Bayes procedure instead allows one to select a single value of the hyper-parameters instead of integrating over all values. This is justified as approximating the hyper-parameter posterior as a point mass centred at the mode:

$$p(\beta|\mathbf{X}) \approx \delta_{\hat{\beta}}(\beta) \text{ where } \hat{\beta} = \arg\max_{\beta} p(\beta|\mathbf{X}) \tag{1.93}$$

If we additionally assume a uniform prior $p(\beta)$, then application of Bayes' rule allows us to reach the equivalent condition $\hat{\beta} = \arg\max_{\beta} p(\mathbf{X}|\beta)$, where $p(\mathbf{X}|\beta)$ is the marginal likelihood.

This point mass approximation means that any operation in which we would be required to marginalise out β becomes computationally easier. One example is if we are interested in the posterior over parameters. Instead of integrating over the entirety of $p(\beta|\mathbf{X})$, only a single value of β ever needs to be considered, and this translates into simply conditioning the posterior on the most likely β:

$$p(\theta|\mathbf{X}) = \int p(\theta|\mathbf{X}, \beta)p(\beta|\mathbf{X}) \, \mathrm{d}\beta \tag{1.94}$$

$$\approx \int p(\theta|\mathbf{X}, \beta)\delta_{\hat{\beta}}(\beta) \, \mathrm{d}\beta \tag{1.95}$$

$$= p(\theta|\mathbf{X}, \hat{\beta}) \tag{1.96}$$

The approximation is often reasonable. If the number of hyper-parameters is small, this can have the consequence that $p(\beta|\mathbf{X})$ is highly peaked. In this scenario, integrating over β can give a similar result to fixing it at $\hat{\beta}$.

Empirical Bayes uses the data to estimate the prior distribution, which is against the Bayesian philosophy. Technically this is no longer a prior in the Bayesian sense, since it does not reflect our belief prior to observing the data.

1.3 Gaussian Processes and Covariance Functions

1.3.1 Gaussian Processes

A Gaussian Process (GP) [see e.g., Rasmussen and Williams, 2005] is a (potentially infinite) collection of random variables such that any finite subset has a multivariate normal joint distribution. The random variables may have any arbitrary index set $\mathcal{X}$, but here we will focus on $\mathcal{X} = \mathbb{R}^d$. The set of random variables is collected into a *random process* f, where a single $\mathbf{x} \in \mathcal{X}$ is used to index a single real-valued random

variable $f(\mathbf{x})$.

A GP can be seen as an infinite generalisation of a multivariate normal distribution, and is specified completely by its mean function $m(\mathbf{x})$ and covariance function $C(\mathbf{x}, \mathbf{x}')$, which are infinite-dimensional versions of the mean vector and covariance matrix. To indicate that we have a GP prior on the function $f(\mathbf{x})$, we write

$$f(\mathbf{x}) \sim \mathcal{GP}(m(\mathbf{x}), C(\mathbf{x}, \mathbf{x}')) \tag{1.97}$$

however, for simplicity and without loss of generality, we will assume $m(\mathbf{x}) = 0$.

By the definition of a GP, the joint distribution of any finite set of random variables $\mathbf{f} = [f(\mathbf{x}_1) \cdots f(\mathbf{x}_n)]^\top$ is a multivariate normal.

$$\mathbf{f}|\mathbf{X} \sim \mathcal{N}(\mathbf{f}|\mathbf{0}, \mathbf{K}) \tag{1.98}$$

where

$$\mathbf{K}_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) \tag{1.99}$$

Here the covariance matrix $\mathbf{K}$ has been specified element-wise, and depends on the choice of covariance function C. The covariance function controls how we expect f to behave, and choosing an appropriate covariance function is important in modelling using GPs. This is further discussed in **Section 1.3.2**. To correctly model a random process, the GP must satisfy *consistency*, meaning that if we expand the set of observed variables, this does not change the distribution of the original set. Mathematically this means adding a single random variable $f(\mathbf{x}^*)$ to the collection $\mathbf{f}$ appends the row/column $[C(\mathbf{x}_1, \mathbf{x}^*) \cdots C(\mathbf{x}_n, \mathbf{x}^*)\, C(\mathbf{x}^*, \mathbf{x}^*)]$ to $\mathbf{K}$.

A GP is a probability distribution over functions, and may be used to specify a prior over the unknown function f. This is a powerful tool in machine learning; one can specify their prior over some mapping (perhaps in a regression context), collect some data and compute a posterior over the mapping. In other contexts, one defines a parametric function to do this and performs Bayesian inference on the parameters. In the GP setting, the prior is placed directly on the function and the GP is *non-parametric*.

Making predictions

Here we show how a GP can be used in a regression context. We may observe the value of f at a finite set of locations and wish to produce a posterior over the value at a new location for which we have no data. Given a set of inputs $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_n]$ paired with observed values $\mathbf{f} = [f(\mathbf{x}_1) \cdots f(\mathbf{x}_n)]^\top$, this amounts

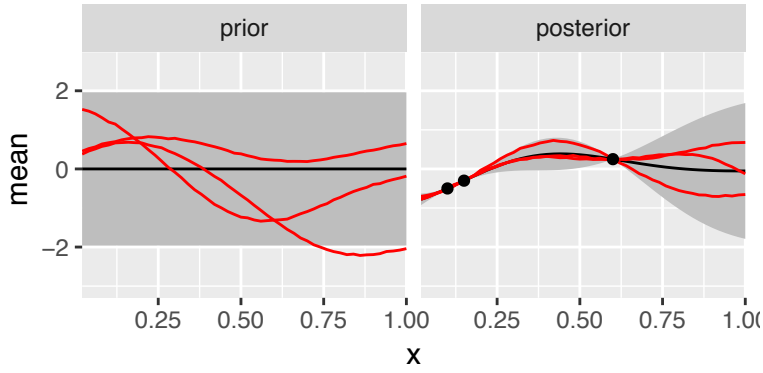


Figure 1.6: *Prior and posterior predictive distributions from a 1-dimensional GP with squared exponential covariance function. Shaded region shows 95% confidence intervals. The prior distribution is the same everywhere: every horizontal slice is a mean zero variance 1 normal. The posterior distribution shows 3 data points which have been conditioned on. The predictive variance shrinks nearer to these points, and returns to the prior far from these points.*

to computing the posterior predictive distribution of a new point $p(f(\mathbf{x}^*)|\mathbf{x}^*, \mathbf{X}, \mathbf{f})$. This can be computed in closed form, which we show starting with the full joint distribution:

$$\begin{bmatrix} \mathbf{f} \\ f(\mathbf{x}^*) \end{bmatrix} |\mathbf{x}^*, \mathbf{X} \sim \mathcal{N}\left(\mathbf{0}, \begin{bmatrix} C(\mathbf{X}, \mathbf{X}) & C(\mathbf{X}, \mathbf{x}^*) \\ C(\mathbf{x}^*, \mathbf{X}) & C(\mathbf{x}^*, \mathbf{x}^*) \end{bmatrix}\right) \tag{1.100}$$

Here we are using the notation where $C(\mathbf{X}, \mathbf{X})$ is an $n \times n$ matrix where the i, jth element is $C(\mathbf{x}_i, \mathbf{x}_j)$. Similarly, $C(\mathbf{X}, \mathbf{x}^*)$ is length-n column vector containing $C(\mathbf{x}_1, \mathbf{x}^*) \cdots C(\mathbf{x}_n, \mathbf{x}^*)$.

Now through applying the Gaussian identity for conditioning on a variable, we obtain the posterior predictive distribution

$$f(\mathbf{x}^*)|\mathbf{x}^*, \mathbf{X}, \mathbf{f} \sim \mathcal{N}(f(\mathbf{x}^*)|0, C(\mathbf{x}^*, \mathbf{x}^*) - C(\mathbf{x}^*, \mathbf{X})C(\mathbf{X}, \mathbf{X})^{-1}C(\mathbf{X}, \mathbf{x}^*)) \tag{1.101}$$

Modelling noisy observations

So far we have assumed access to the true function values $f(\mathbf{x})$, meaning the regression like will pass exactly through all collected data points. A more realistic modelling scenario is that we have noisy observations y, which we model with the Gaussian likelihood

$$y|f(\mathbf{x}), \sigma_n^2 \sim \mathcal{N}(y|f(\mathbf{x}), \sigma_n^2) \tag{1.102}$$

where we now have a parameter σ_n^2 describing the variance on the i.i.d observation noise. In the special case of this Gaussian likelihood, we may integrate out the latent values $f(\mathbf{x})$ using our Gaussian prior (**Equation 1.98**) to obtain the marginal likelihood in closed form:

$$\mathbf{y}|\mathbf{X}, \sigma_n^2 \sim \mathcal{N}(\mathbf{y}|\mathbf{0}, C(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I) \tag{1.103}$$

Comparing this with **Equation 1.98**, the only difference is that $C(\mathbf{X}, \mathbf{X})$ has changed to $C(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I$. This change may also be obtained by switching from a base covariance function $C(\mathbf{x}, \mathbf{x}')$ to a 'noisy' covariance function $C_{\text{noisy}}(\mathbf{x}, \mathbf{x}') = C(\mathbf{x}, \mathbf{x}') + \sigma_n^2 \mathbb{1}[\mathbf{x} = \mathbf{x}']$. This also guarantees the invertibility of $\mathbf{K}$, since many covariance functions produce a semi-definite $\mathbf{K}$ and adding σ_n^2 to the diagonal ensures all eigenvalues are positive. The value of σ^2 may be selected by maximising the marginal likelihood.

Now, given observations $\mathbf{X}$, $\mathbf{y}$, and using the same logic as above, our posterior predictive distribution at a new point $\mathbf{x}^*$ becomes:

$$\begin{aligned} f(\mathbf{x}^*)|\mathbf{x}^*, \mathbf{X}, \mathbf{y}, \sigma_n^2 \sim \mathcal{N}(&C(\mathbf{x}^*, \mathbf{X}) \left(C(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I\right)^{-1} \mathbf{y}, \\ &C(\mathbf{x}^*, \mathbf{x}^*) - C(\mathbf{x}^*, \mathbf{X}) \left(C(\mathbf{X}, \mathbf{X}) + \sigma_n^2 I\right)^{-1} C(\mathbf{X}, \mathbf{x}^*)) \end{aligned} \tag{1.104}$$

Here we are predicting $f(\mathbf{x}^*)$ as opposed to y^* since we would like to predict the true latent value, not the noisy observation.

1.3.2 Covariance Functions

A GP requires a covariance function which defines the set of functions the GP is able to represent. A valid covariance function is any function which, when constructing a matrix of pairwise evaluation between elements of the index set, always produces a positive semi-definite matrix. This means that the $d \times d$ matrix $\mathbf{K}$ where

$$(\mathbf{K})_{ij} = C(\mathbf{x}_i, \mathbf{x}_j) \quad \forall\, i, j \in 1 \cdots d \tag{1.105}$$

is always a valid covariance matrix.

Covariance functions are closed under summation and multiplication. This allows the construction of new covariance functions from old. Additionally, many covariance functions are easy to interpret. A consequence of this is that they are an attractive way of specifying a prior. A practitioner may be able to encode their prior knowledge of some process into a covariance function, thus producing a prior distribution reflecting their own understanding. Compositionality and interpretability are

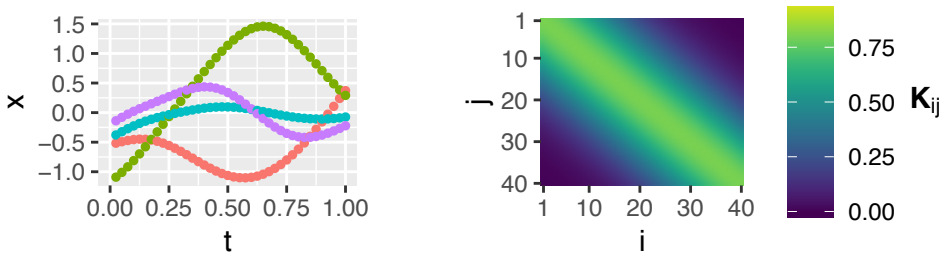


Figure 1.7: *Squared Exponential. Parameter values:* $\sigma_n^2 = 0.8, \ell = 0.3$. *Left: 4 samples. Right:* $\mathbf{K}$.

taken advantage of by Lloyd et al. [2014] to produce human-readable descriptions of data trends by fitting a GP with a compound covariance function.

A number of widely-known covariance functions are briefly discussed here. To illustrate the covariance structure they encode, samples from $\mathcal{N}(\mathbf{0}, \mathbf{K})$ are shown. For more detail consult Rasmussen and Williams [2005, Chapter 4].

We work mostly with stationary covariance functions. These depend only on the distance $r := ||\mathbf{x}_i - \mathbf{x}_j||$, and are translation invariant. To keep notation compact, we show these as a function of r below.

Squared Exponential

$$C_{\text{SE}}(r) = \sigma_s^2 \exp\left(-\frac{r^2}{2\ell^2}\right) \tag{1.106}$$

The Squared Exponential (SE) covariance function is an infinitely differentiable function of r, which is reflected in the smooth looking samples in **Figure 1.7**. σ_s^2 is the "signal variance", which gives the variance at any single x_i. The length scale, ℓ, controls how rapidly the covariances decay as the distance r increases.

Rational Quadratic

$$C_{\text{RQ}}(r) = \sigma_s^2 \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha} \tag{1.107}$$

The Rational Quadratic (RQ) covariance function is a scale mixture of SE covariance functions with different length scales. α controls the heaviness of the tail of C_{RQ}; α closer to 0 encodes longer range covariances. This can be seen in the covariance matrix produced; near $r = 0$ the covariance looks like the SE, but as r increases, the covariance dies off much more slowly. In the limit $\alpha \to \infty$, the RQ becomes the SE, so if the RQ is parametrised such that this limit can be reached, using the RQ produces a space of models which includes all those which can be

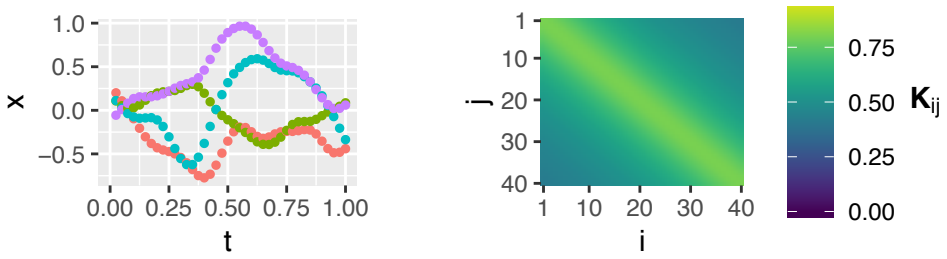


Figure 1.8: *Rational Quadratic. Parameter values* $\sigma_n^2 = 0.8, \ell = 0.3, \alpha = 0.2$. *Left: 4 samples. Right:* $\mathbf{K}$.

reached using the SE.

Independent

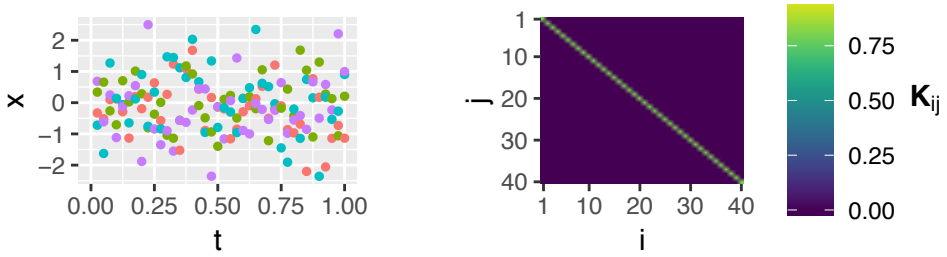


Figure 1.9: *Independent. Parameter value* $\sigma_n^2 = 0.8$. *Left: 4 samples. Right:* $\mathbf{K}$.

$$C_{\text{ind}}(r) = \sigma_n^2 \mathbb{1}[r = 0] \tag{1.108}$$

The independent covariance function encodes that the values at each x_i are independent of each other, and independent of their locations $\mathbf{x}_i$. The covariance matrix produced is a scaled identity matrix $\sigma_n^2 I$, and thus the samples in **Figure 1.9** look like white noise.

Melkumyan-Ramos

$$C_{\text{MR}}(r) = \begin{cases} \sigma_s^2 \left[\frac{2+\cos(2\pi r l^{-1})}{3}(1 - \frac{r}{l}) + \frac{1}{2\pi}\sin(2\pi\frac{r}{l}) \right] & \text{if} \quad r < l \\ 0 & \text{if} \quad r \geq l \end{cases} \tag{1.109}$$

The Melkumyan-Ramos (MR) covariance function [Melkumyan and Ramos, 2009], which we have named after the authors, is a *compactly supported covariance function* and is equal to exactly 0 whenever r is greater than the hyper-parameter l. This produces a sparse $\mathbf{K}$, which has computational advantages in storage and

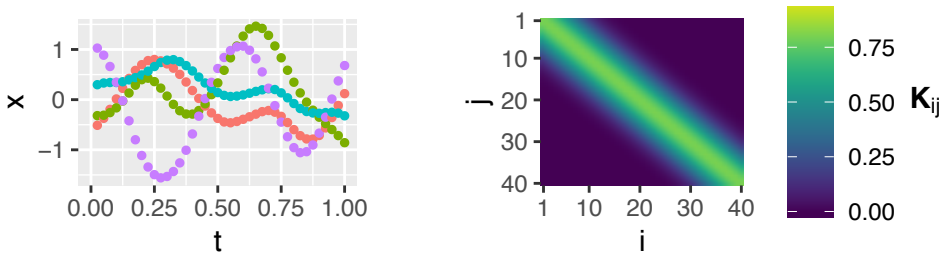


Figure 1.10: *MR covariance function [Melkumyan and Ramos, 2009]. Parameter values* $\sigma_n^2 = 0.8, l = 0.5$. *Left: 4 samples. Right:* $\mathbf{K}$.

inversion.

The MR may be obtained by taking a single period of a sine wave and shifting and scaling such that its peak value σ_s^2 is at 0 and it touches 0 at $\pm r$. Close to $r = 0$, the MR approximates the SE covariance function, producing visually similar samples.

Positive Definite Covariance Functions

All of the above covariance functions are valid in the mathematical sense of positive-semi-definiteness, and they will always produce positive-semi-definite covariance matrices. However, the semi-definiteness can be a problem computationally. A Gaussian likelihood requires inverting the covariance matrix, and this cannot be performed if the covariance matrix is only semi-definite.

The GP literature confronts the same problem. When fitting a Gaussian process, it is standard to take the semi-definite covariance matrix $\mathbf{K}$, and add σ_n^2 to each element on the diagonal. σ_n^2 is a another parameter which may be learned, and has the interpretation of the variance of i.i.d noise which is added to some true latent function value f_i.

Adding σ_n^2 to the diagonal of $\mathbf{K}$ is equivalent to adding the independent covariance function C_{ind} to the selected covariance function C, producing the new covariance function C_{noisy}:

$$C_{\text{noisy}}(\mathbf{x}_i, \mathbf{x}_j) = C(\mathbf{x}_i, \mathbf{x}_j) + C_{\text{ind}}(\mathbf{x}_i, \mathbf{x}_j) \tag{1.110}$$

$$= C(\mathbf{x}_i, \mathbf{x}_j) + \sigma_n^2 \mathbb{1}[\mathbf{x}_i = \mathbf{x}_j] \tag{1.111}$$

This also adds σ_n^2 to each of the eigenvalues of $\mathbf{K}$, guaranteeing a positive-definite K with a smallest eigenvalue of at-least σ_n^2.

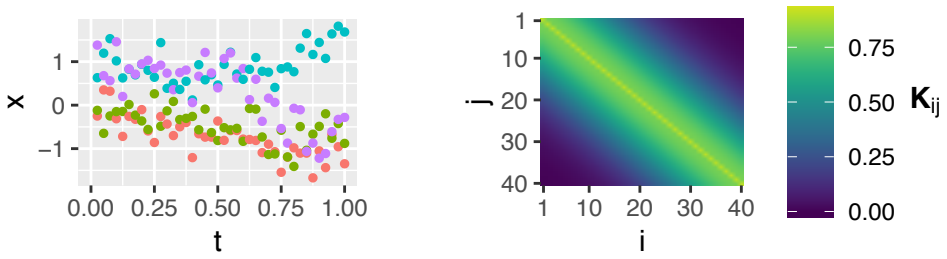


Figure 1.11: *Noisy Squared Exponential. Parameter values:* $\sigma_n^2 = 0.8, \ell = 0.3, \sigma_n^2 = 0.1$. *Left: 4 samples. Right:* $\mathbf{K}$.

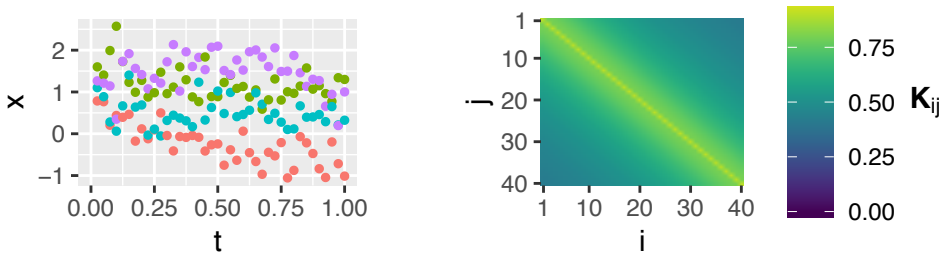


Figure 1.12: *Noisy Rational Quadratic. Parameter values* $\sigma_n^2 = 0.8, \ell = 0.3, \alpha = 0.2, \sigma_n^2 = 0.1$. *Left: 4 samples. Right:* $\mathbf{K}$.

Noisy Squared Exponential

$$C_{\text{NSE}}(r) = \sigma_s^2 \exp\left(-\frac{r^2}{2\ell^2}\right) + \sigma_n^2 \mathbb{1}[r = 0] \tag{1.112}$$

The Noisy Squared Exponential covariance function is simply a sum of the SE and Independent covariance functions. For small values of σ_n^2, this will be a close approximation to the SE covariance function, but will produce an invertible $\mathbf{K}$. The addition of σ_n^2 to the diagonal can be observed in the right-hand plot of **Figure 1.11**, which is otherwise identical to **Figure 1.7**.

Noisy Rational Quadratic

$$C_{\text{NRQ}}(r) = \sigma_s^2 \left(1 + \frac{r^2}{2\alpha\ell^2}\right)^{-\alpha} + \sigma_n^2 \mathbb{1}[r = 0] \tag{1.113}$$

Similarly to the Noisy SE, the Noisy RQ is a covariance function that may be used in practise due to always producing positive definite covariance matrices as long as $\sigma_n^2 > 0$.

1.4 Summary

This introductory chapter has covered mathematical and statistical background core to this thesis. A selection of latent variable models have been surveyed in **Section 1.1**, each illustrating a key way in which latent variable models may differ. Common techniques for performing inference in such models were introduced in **Section 1.1.2**.

Section 1.2 introduced the Bayesian statistical background. Selection of a model from a candidate set in a Bayesian setting was covered, introducing Bayesian Occam's razor, Bayes' Factors and the Laplace approximation.

Finally, **Section 1.3** introduced the Gaussian Process in a modelling context. It is shown how the GP may be used as a Bayesian prior over a function, where prior belief over properties of the function may be expressed via choice of a covariance function. A number of covariance functions were illustrated.

Chapter 2

Artificial Olfaction

A major theme of this thesis is statistical methodology in artificial olfaction. The previous chapter introduced the required statistical background, and this chapter introduces artificial olfaction in three major parts: background material, techniques specific to our lab, and a series of analyses performed by the author.

Our focus is on the use of artificial olfaction in a medical setting. A core task in this context is to analyse the odour of a patient sample – e.g., urine, breath, blood, stool, sputum or swab – and determine if the patient has a disease of interest, or to distinguish between similar diseases. Such a tool could be of value medically, enabling a rapid, non-invasive procedure able to aid diagnoses or track disease progression.

Section 2.1 introduces some background, covering biological olfaction, olfaction in medicine, and existing work. **Section 2.2** describes three methods for artificial olfaction used in the novel scientific content of this thesis. These are the Electronic Nose (E-nose), Field Asymmetric Ion Mobility Spectrometry (FAIMS), and Gas Chromatography-Ion Mobility Spectrometry (GC-IMS). **Section 2.3** contains data analyses performed by the author for this thesis.

2.1 Background

Artificial Olfaction instruments are a class of devices which are, in a certain sense, able to "smell". Some, like the Electronic Nose, are inspired directly by mammalian olfaction, whilst others work under a very different paradigm. A major property that they all share is that they analyse a gas of interest and produce a high dimensional numerical output which characterises the odour of the gas. The instruments need not be able to identify compounds which make up the gas; it is only required that similar

gasses produce similar outputs. When paired with statistical and machine learning techniques, an artificial olfaction instrument may be to identify some property of interest, such as the presence of a disease in a hospital patient.

Olfaction has potential value in a medical setting for diagnosis and monitoring. However, relying on a human to identify a disease by smell would be highly variable, depending on the state of health, level of training and any strong tasting foods eaten. Using a trained animal introduces further issues of of communication difficulties, as well as health hazards when brought in to a hospital. A quantitative measurement using some analytical instrument would be the ideal solution, which techniques in artificial olfaction offer.

Artificial Olfaction is an attractive piece of instrumentation for medical use. If able to perform diagnosis, such a machine can be non-invasive, only requiring patient urine, breath or other readily supplied sample. Diagnosis also has the potential to be rapid, since many artificial olfaction devices are portable enough to be used at the point of care, and only take seconds to minutes to provide a measurement.

2.1.1 Biological Olfaction

Biological olfaction in humans [e.g., Gardner and Bartlett, 2000, Chapter 3] begins in the olfactory epithelium—a patch of tissue around 6 cm^2 containing around 10 million olfactory receptor neurons [Schacter et al., 2014]. Each receptor neuron has 10–60 hair-like cilia, the membranes of which contain olfactory receptor (OR) proteins. There are around 400 types of OR protein in humans [Gilad and Lancet, 2003], each of which is broadly sensitive to a wide range of compounds. Upon interacting with an odourant, an OR undergoes a conformation change and leads to an action potential in the OR neuron, carrying information to the olfactory bulb in the brain. This is illustrated in **Figure 2.1a**.

The broad specificity of each OR is important; a given compound likely binds to a number of ORs, and an OR will be activated by a number of compounds. This means that odourants that have not been previously encountered will still likely produce an activation pattern which is distinct from known odours. **Figure 2.1b** illustrates a response pattern across each sensor type which uniquely identifies a given odour.

The olfactory epithelium is coated in a mucous layer approximately $20\mu m$ thick, which introduces two effects. Firstly, the interaction between OR and odourant occurs in an aqueous environment. Secondly, different molecules may take different times to cross the mucous layer, producing a chromatographic affect. This temporal separation may carry information to aid the discrimination of odours, and

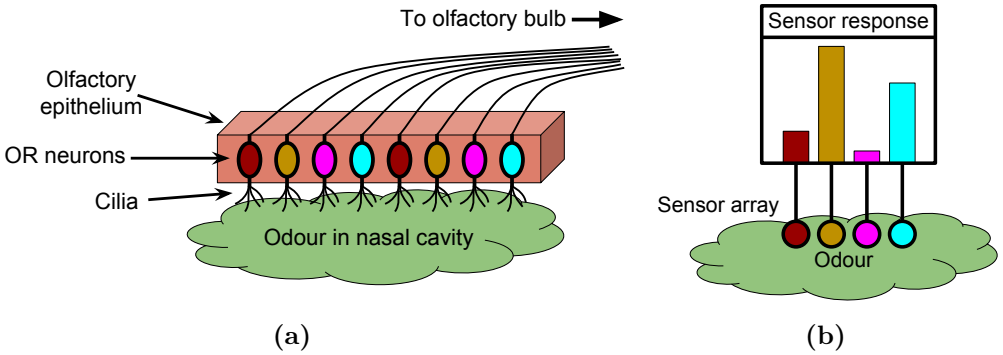


Figure 2.1: ***(a)*** *Olfactory Receptor (OR) neurons in the olfactory epithelium end in cilia which enter the nasal cavity. These cilia are covered in broadly sensitive ORs, and an odourant molecule binding to such a receptor will contribute towards inducing an action potential in the OR neuron. This action potential is carried to the olfactory bulb in the brain. There are approximately 400 unique types of OR neuron in humans [Gilad and Lancet, 2003], each with different ORs on the cilia.* ***(b)*** *A simplified cartoon of human olfaction. A given odour will produce an activation pattern across all sensors. Different odours will produce different activation patterns. The pattern in its entirety characterises the smell.*

has been applied in an electronic nose setting in Che Harun et al. [2012].

The olfactory neurons continuously die and are regenerated with a half-life of around 4–8 weeks. Most artificial olfaction techniques have a fixed sensor or set of sensors over their lifetime, and thus may suffer from sensor drift.

2.1.2 Medical Diagnosis

Use of artificial olfaction in medicine is predominantly limited to scientific studies, but good discriminative ability has been shown for a variety of diseases using breath [Brinkman et al., 2017; Lewis et al., 2017; Montuschi et al., 2013; Dragonieri et al., 2007; Montuschi et al., 2010; DAmico et al., 2010; Dragonieri et al., 2009; Kolk et al., 2012; Bruins et al., 2013], stool [van Gaal et al., 2017; Bomers et al., 2015], sputum [Kolk et al., 2010], and urine [Westenbrink et al., 2015; Arasaradnam et al., 2015, 2014, 2013]

There is a long history of using smell in medicine. Early medical practitioners were aware that the odour of bodily excretions can be influenced by the presence of disease. Starting from 400BC, human analysis of the odour of sputum would be aided by vaporising the sputum on hot coals [Wilson and Baietto, 2011]. From the early 19^{th} century descriptive aromas of diseases have been published to aid with diagnosis, of which we present a small subset in **Table 2.1**.

There are many non-human animals with a far superior sense of smell. In

Disease/Disorder	Body source	Descriptive aroma
Acromegaly	Body	Strong, offensive
Anaerobic infection	Skin, sweat	Rotten apples
Azotemia (prerenal)	Urine	Concentrated urine odour
Bacterial proteolysis	Skin	Over-ripe Camembert
Bacterial vaginosis	Vaginal discharge	Amine-like
Bladder infection	Urine	Ammonia
Bromhidrosis	Skin, nose	Unpleasant
Dariers disease	Buttocks	Rank, unpleasant odour

Table 2.1: *Descriptive aromas of a variety of diseases. First 8 items of a 50 item table. Copied from Wilson and Baietto [2011].*

2010, a systematic review of detection of human cancers by canines concluded that dogs may be able to smell cancer with a high enough sensitivity and specificity to be diagnostically useful [Moser and McCulloch, 2010].

The reasons for changes in odour of biological samples due to disease are numerous and complex, depending on the condition at hand. Volatile organic compounds (VOCs) are produced by pathogens at a site of infection, as well as by the inflammatory and immune responses by the host [Covington et al., 2015]. Many gastrointestinal diseases cause, or are caused by, imbalances in gut microbial colonies, leading to production of different VOCs [Sagar et al., 2015]. Many volatiles produced in disease are excreted in sweat, urine, breath, blood, sputum and other samples which enable analysis.

There is ample evidence for the existence of diagnostically useful signal in the odour of patient breath, sputum, stool and other biological samples. However, extracting this signal is challenging. There is a great deal of natural variation in odour (e.g., smoking, BMI, and gender [Blanchet et al., 2017]), and this uninformative 'noise' may swamp the useful signal. The magnitude of the interesting signal may also be smaller than variation between experimental batches or sample degradation whilst in storage [Esfahani et al., 2016]: Berkhout et al. [2016] found the VOC profiles of stool samples to be affected by sample mass, sample temperature, sample dilution, freeze-thaw cycles, and time at room-temperature storage.

Industry has begun to show interest in VOC analysis for medical purposes. For example, IMSPEX produce the BreathSpec™, which is a purpose built GC-IMS for analysing human breath in which a flow-controlled mouthpiece allows direct sampling of breath[1]. Owlstone have also begun a medical venture[2], using their

[1] `https://breathspec.com`
[2] `https://www.owlstonemedical.com`

ReCIVA Breath Sampler as part of a trial for lung cancer detection.

2.2 Instruments

In the analyses done for this thesis, three different instruments were used for artificial olfaction: Electronic Nose (E-nose), Field Asymmetric Ion Mass Spectrometry (FAIMS), and Gas Chromatography-Ion Mobility Spectrometry (GC-IMS). Each machine detects different properties of a gas, and has relative advantages and disadvantages.

2.2.1 The Electronic Nose

The Electronic Nose (E-nose) [e.g., Gardner and Bartlett, 2000] was invented in the 1980s as a machine to discriminate odours in a manner inspired by biological olfaction [Persaud and Dodd, 1982]. The key idea is to measure a gas mixture by using multiple diverse sensors, each sensitive to a broad range of volatile compounds, much like the human nose. Different mixtures produce different response patterns across all sensors, known in the literature as a "smell signature". In a medical context, a signal of interest (such as presence of a disease) will likely change the concentrations of many different VOCs. Furthermore, each VOC will likely bind to a number of sensors. The relevant signal will thus likely be mixed up across sensors. Identification of a relevant signal in a noisy background will be difficult for a human operator, but amenable to a machine learning approach.

The gold standard analytical chemistry technique for characterising a gas is GC-MS. The E-nose differs from this in that it is not currently possible to determine the molecular content of the gas mixture. Using the E-nose is comparatively inexpensive and requires less training than the GC-MS, and may also use ambient air as a carrier gas [Fens et al., 2012].

An E-nose requires an odour delivery system, allowing controlled delivery of a gas to a sensor chamber. In this thesis we are interested in the case where samples of interest (such as blood, stool or urine) have been prepared into vials in consistent quantity and dilution. This lends itself to a headspace sampling delivery system. Here, the prepared samples are left until the gas above the sample (the "headspace") has equilibrated, and a sample of the headspace is delivered to the sensor chamber via syringe. Hand delivery via syringe is possible, but an automated system (an autosampler, illustrated in **Figure 2.2**) can improve repeatability through consistent syringe operation and timing. Alternative systems may use flow injection, where a

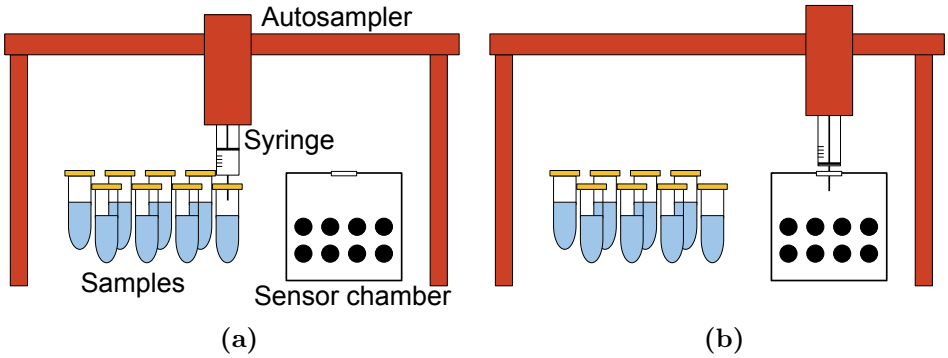


Figure 2.2: *The Electronic Nose may take advantage of an auto-sampler, providing an automated and finely controlled system of delivering a sample headspace to an E-nose sensor array.* ***(a)****: Sampling the headspace of a prepared sample using a syringe.* ***(b)****: Injecting the headspace sample into the sensor chamber.*

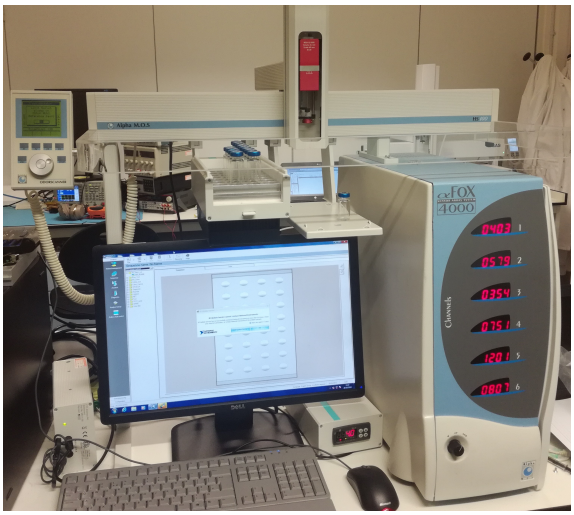


Figure 2.3: *The alpha-MOS FOX-4000 Electronic nose with auto-sampler.*

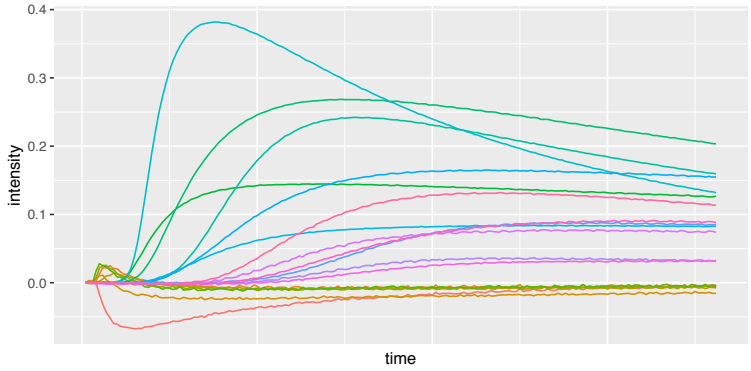


Figure 2.4: *Typical Alpha-MOS FOX-4000 measurement output. Each curve is a time-series of the response of a single sensor as a gas is passed through the sensor array.*

carrier gas is passed over a sample of interest and across the sensors, but this is not as appropriate in this setting.

Many types of sensor—such as MOS sensors—interact chemically with the gas being measured, and may remain contaminated after the sample is removed. For this reason it is common to decontaminate the E-nose using purified air after each measurement is taken. The amount of time required for decontamination will depend on the sample being analysed: certain compounds may interact strongly with the sensors and require many hours to days for the contamination levels to drop to an acceptably low level. Chemical interactions causing permanent changes to the sensors may cause *sensor drift* to occur over periods of months to years, changing the response characteristics of the sensor.

Many different sensors have been used in the electronic nose [James et al., 2005]. These can differ in their recovery times and response to different compound groups. Sensors are often metal oxide semiconductors, conducting polymers or piezoelectric sensors, but can be more exotic: Rains et al. [2006] use live wasps trained to react to specific substances.

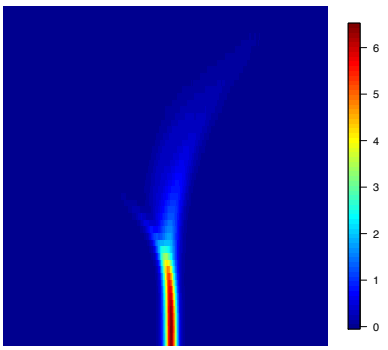
The E-nose has been applied successfully in analysing patient urine to detect colorectal cancer [Westenbrink et al., 2015], to discriminate between IBD subtypes and healthy controls [Arasaradnam et al., 2013], and to identify urinary tract cancers [Bernabei et al., 2008].

The E-nose used in this thesis is the Alpha-MOS FOX-4000 (**Figure 2.3**), which has Metal Oxide based sensors. An HS100 auto-sampler is included, allowing fine control over the amount of headspace sampled, as well as the flow of headspace into the sensing unit. **Figure 2.4** shows typical data representing a single sample from the FOX-4000; a time-series of the response of 18 sensors as a gas is passed over the sensor array.

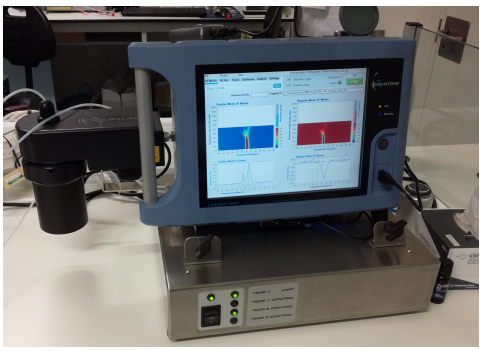
2.2.2 Field Asymmetric Ion Mobility Spectrometry

Another instrument used for artificial olfaction is the Field Asymmetric Ion Mobility Spectrometer (FAIMS) [Guevremont, 2004; Owlstone Nanotech, 2006; Wilks et al., 2012; Shvartsburg et al., 2009]. Unlike the electronic nose which uses multiple heterogeneous chemical sensors, the FAIMS uses a single ion detector. The molecules making up a gas are ionised, and these ions are separated out according to their mobility, producing a spectrum. This spectrum is produced under different intensities of separation, producing a two-dimensional separation as illustrated in **Figure 2.5a**.

We use the Owlstone Lonestar (**Figure 2.5b**) [Owlstone Nanotech, 2006;



(a) *Example FAIMS output. The data are non-negative, 26112 dimensional and fit on a* 512×51 *grid.*



(b) *FAIMS photograph. Sample is housed and analysed in the black section on the left. Bulk of machine is data display and carrier gas regulator.*

Figure 2.5: *Field Asymmetric Ion Mass Spectrometry (FAIMS)*

Wilks et al., 2012; Shvartsburg et al., 2009] for the analyses in this thesis. Ionisation, ion separation and detection are performed on a single fingertip sized chip [Owlstone Nanotech, 2006]. The bulk of the machine is for housing the sample and circuitry, maintaining a constant flow of carrier gas, and displaying the output.

A number of medical studies have been performed to assess the ability of FAIMS to distinguish a set of diseased patients from a matched control group, as reviewed by Covington et al. [2015]. Diseases investigated with good results include: Hepatic Encephalopathy using breath [Arasaradnam et al., 2016a], Tuberculosis using breath [Sahota et al., 2016], Inflammatory Bowel Disease using breath [Arasaradnam et al., 2016b] and stool [van Gaal et al., 2017], Non alcoholic fatty liver disease using urine [Arasaradnam et al., 2015], C. difficile infection using stool [Bomers et al., 2015], and Colorectal cancer using urine [Arasaradnam et al., 2014].

For analysing a liquid sample, we use a 5ml liquid sample prepared in a vial, which is inserted into an airtight chamber. The chamber is heated such that the sample releases gas into the "headspace"; the volume directly above the liquid sample containing the gas to be analysed. This headspace is analysed by FAIMS. Such a process can be used to measure patient urine, blood, and stool. Patient breath may also be measured, and one way of achieving this is to capture patient breath within a sterile Tedlar bag. This sample can then be cooled and transported, and drawn into the FAIMS using air as a carrier gas [Arasaradnam et al., 2016a].

How it works

The FAIMS relies on *mobility*: a physical property of a charged particle (an ion) subject to an electrical field whilst travelling through a gas. The mobility μ is

defined as the constant of proportionality relating the strength of the electric field E to the terminal drift velocity v_d:

$$v_d = \mu E \tag{2.1}$$

At low electric fields, mobility may be approximated as being independent of the field strength. However in high electric fields the dependence on E becomes important. The dependence of μ on E is different per ionic species. FAIMS works in the high electric field regime, and uses these properties to separate out ionic species.

The sample headspace is ionised upon entering the FAIMS, which is commonly done using ultra-violet light or the radioactive isotope Ni63. An electrical field then drives these ions between two metal plates towards an ion detector. As the ions travel towards the detector, they are subject to an electric field orthogonal to the direction of travel which is made up of a constant and a periodic component.

The periodic component of the applied field is known as the Dispersion Field (DF). This has an asymmetric waveform which is briefly, strongly positive, and slightly negative for a longer period of time. The area (voltage $\times$ time) in each phase is the same. Due to the species-dependent change in ion mobility in strong electric fields, the ratio of mobilities between the positive and negative phase will be different for each ionic species. This will cause a net drift towards either of the metal plates for nearly all ionic species. If an ion touches a plate, its charge is lost and it will not be detected. If the ratio of mobilities is approximately 1, the ion will continue though the filter and be detected. By increasing the strength of the DF, the ions are dispersed more strongly. This will also change the mobility ratios of all species due to the non-linearity of mobility in high electrical potentials. This process is illustrated in **Figure 2.6**.

A DC component to the electrical field is also added, known as the Compensation Voltage (CV), which introduces a net drift for all ions. Varying the CV for a fixed DF allows different ionic species with mobility ratios away from 1 to reach the detector.

By cycling through a range of values for both the DF (strictly positive) and CV (symmetric around 0V) and measuring the ionic flux, the output in **Figure 2.5a** is produced. The x and y axes are CV and DF respectively, and the value at each location is ionic flux, which must be non-negative. The image is 26112 dimensional, since to produce the output the FAIMS steps through 512 CV values and 51 DF values.

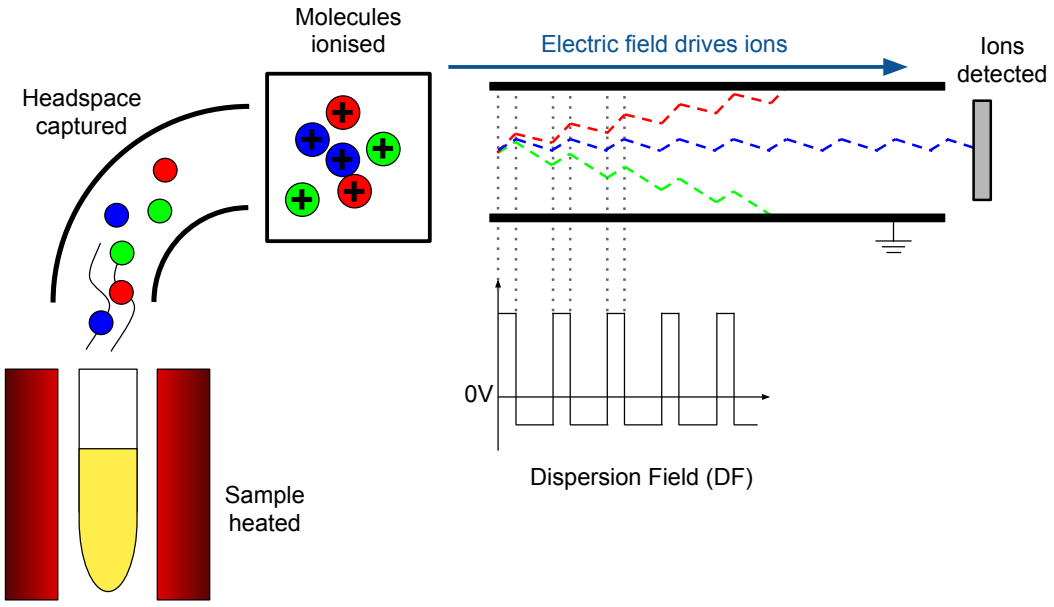


Figure 2.6: *The FAIMS works by heating the sample, then collecting and ionising the headspace. The ions are dragged between two metal plates by an electric field. An asymmetric electrical field waveform is applied, giving each ionic species a net drift towards the top or bottom plate. Ions that touch either plate lose their charge and are not detected. Ions with no net drive reach the end of the filter and are detected. This is repeated for 512 Compensation Voltages and 51 Dispersion Voltages [Guevremont, 2004].*

Properties of the data

Typically a single sample is left in the FAIMS long enough to produce three measurements. These measurements are not identical because the sample will continue to be heated whilst in the FAIMS, changing the composition of the headspace. Certain compounds can also be exhausted, not appearing in the third run, and others can take longer to dissolve out of the liquid sample.

An advantage of FAIMS over E-nose is reduced susceptibility to sensor drift. Sensor contamination can still be an issue; for example, after a sample has been run through the FAIMS, the sensor can remain contaminated for a period of time depending on the contaminating compound and concentration. Running a blank sample for a few runs can remove this contamination, but certain compounds such as high percentage alcohol can contaminate the FAIMS for hours or days.

The output of the FAIMS will depend not only on the sample but also the background smell of the room. By subtracting the third from the first run of a single sample, this can be somewhat compensated for as only the differences are retained. This can also help remove contamination from a previous sample from the signal.

Algorithm 2.1: Statistical pipeline pseudo-code

Input: Data $\mathbf{X}$, labels $\mathbf{y}$, parameter σ

$\mathbf{X}_{\text{wav}} \leftarrow \texttt{Wavelet2D}(\mathbf{X})$

Drop columns of $\mathbf{X}_{\text{wav}}$ with standard deviation $< \sigma$

cross-validate

  Split $\mathbf{X}_{\text{wav}}, \mathbf{y}$ into $(\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}}), (\mathbf{X}_{\text{te}}, \mathbf{y}_{\text{te}})$

  $\texttt{pvals} \leftarrow \texttt{wilcox}(\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$

  Keep columns of $\mathbf{X}_{\text{tr}}, \mathbf{X}_{\text{te}}$ corresponding to the two smallest `pvals`

  Train classifiers with $(\mathbf{X}_{\text{tr}}, \mathbf{y}_{\text{tr}})$

  Predict $\mathbf{y}^*_{\text{te}}$ from $\mathbf{X}_{\text{te}}$ on each classifier

end

Concatenate each $\mathbf{y}^*_{\text{te}}$ into $\mathbf{y}^*$

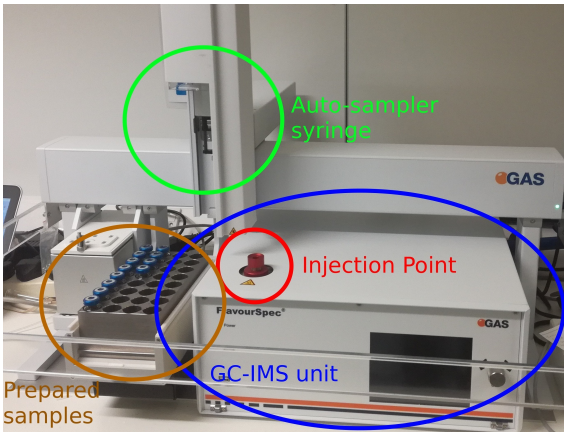
return $\texttt{auc}(\mathbf{y}, \mathbf{y}^*)$

Prediction pipeline

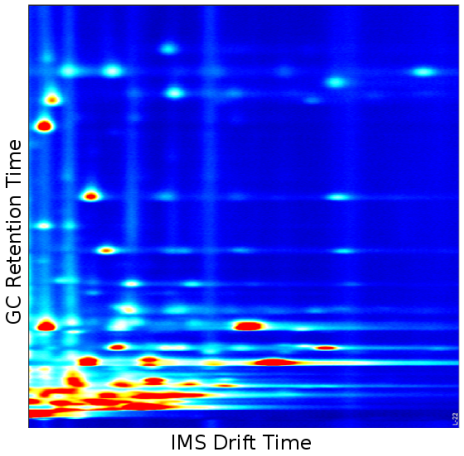
A statistical pipeline has previously been developed (not by the author) to produce cross-validated AUCs from labelled datasets of patient samples as measured by FAIMS [Martinez-Vernon et al.]. Pseudo-code of the pipeline is given in **Algorithm 2.1**.

The pipeline first performs a 2d wavelet transformation independently on each sample, which fits the natural 2d representation of a FAIMS measurement. The standard deviation of each wavelet coefficient is computed across all samples, and wavelets with a standard deviation below a user-specified threshold σ are dropped. This results in ignoring regions of the FAIMS plane which vary little between samples. Wavelets representing corner regions are usually dropped at this stage due to negligible ionic flux.

Supervised feature selection and class prediction is then performed within a cross-validation. For a k-fold cross-validation, within each fold every kth sample is taken to be in the test set. The Wilcoxon rank-sum test [e.g., DeGroot and Schervish, 2013] is used to independently assess each feature (wavelet coefficient) in the test set for whether the distributions of positive and negative samples have the same median. The two features with the smallest p-values are retained and the rest discarded, where this value of two has been selected to maximise performance on past data. Each of a set of classifiers is then trained on the training set, and predictions are made for the class of each sample in the test set. Over all of the folds, a prediction is made for the class of every sample. An AUC is then computed using the known classes and predicted classes.



(a) *Photograph of the Flavourspec GC-IMS with auto-sampler.*



(b) *2d separation of a single sample. Used with permission from olive oil study*[3].

Figure 2.7: *Gas Chromatography-Ion Mass Spectrometry (GC-IMS)*

2.2.3 GC-IMS

GC-IMS combines two well-known analytical chemistry techniques: Gas Chromatography (GC) followed by Ion Mobility Spectrometry (IMS). The GC stage performs a 1d separation of a gas, which then feeds into an IMS system that performs a second separation and detection. A 2d separation is thus produced.

The GC-IMS used in this thesis is the Flavourspec, produced by G.A.S[4], a member of the IMSPEX group. The Flavourspec has been used extensively food product analysis [Fernandez et al., 2017; Márquez-Sillero et al., 2014; Krisilova et al., 2014; Garrido-Delgado et al., 2012]. The setup includes an auto-sampler (visible in **Figure 2.7a**), facilitating more repeatable and less labour-intensive experimentation.

An example Flavourspec output is given in **Figure 2.7b**. The colour is ionic flux at the detector (red high, blue low). The y-axis is the time taken for the detected ions to have passed through the GC column. The x-axis is the time taken for a small packet of ions already having passed through the GC column to pass through the IMS. Individual compounds tend to form visible peaks.

[3] `http://imspex.com/wp-content/uploads/Application_N_olive-oils_N_dstw_130606.pdf`
[4] `www.gas-dortmund.de`

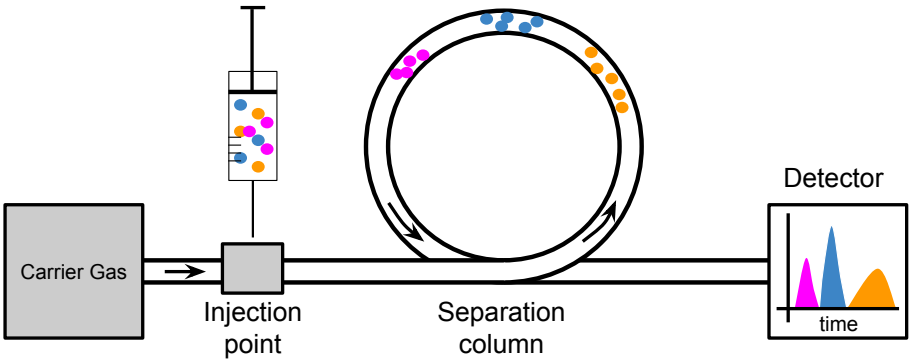


Figure 2.8: *Gas Chromatography: A gaseous analyte is delivered into the injection point by syringe. An inert carrier gas pushes the analyte through a separation column. Compounds within the analyte react with the surface of the separation column to a varying degree, slowing their rate of travel through the column. A detector measures the amount of compound exiting the separation column. Different peaks in the detector correspond to different compounds.*

Gas Chromatography

Gas Chromatography [e.g., Fowlis, 1995] is used in this thesis to produce a pre-separation of the analyte before being separated again by the IMS. GC is made up of three major stages; an injection point, a separation column and a detector. An analyte is introduced through the injection point and carried through the column by a carrier gas. Different compounds in the analyte will take different amounts of time to pass through the column due to interactions with a coating on the column surface. A detector lies at the end of the column, and a *chromatograph* can be produced by plotting the detector response against the period of time following injection. Typically the chromatograph will show a number of peaks each corresponding to a different compound.

An auto-sampler is used to deliver samples of headspace to the GC. A syringe injects a thin band of headspace through an injection point. Although a thinner band enables greater separation resolution, this reduces the amount of analyte available. An inert carrier gas then carries the analyte through the separation column.

The column is made up of glass or metal tubing coated in a thin layer of some liquid or polymer with which different compounds in the analyte will react. Some compounds will react strongly, sticking to the wall of the column, whilst other compounds will remain gaseous. During separation, the column thus contains a gas moving towards the detector (the "mobile phase"), and a solid or liquid coating (the "stationary phase"). Compounds which do not react with the stationary phase will exit the column first and be detected separately to those exiting later.

Certain analytes can contaminate or permanently change the properties of

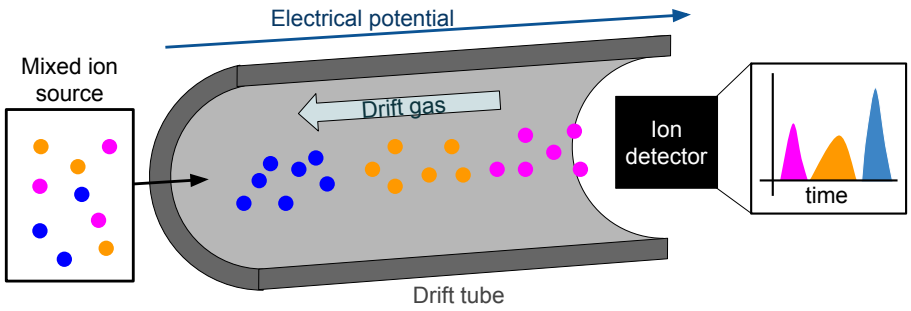


Figure 2.9: *Ion Mobility Spectrometry: Ions from some mixed ion source are released, and an electrical potential drives these ions down a drift tube to an ion detector. The ions drift through a carrier gas which flows in the reverse direction, removing any negatively charged or non-charged particles. Different ion species drift at different rates, and thus reach the detector at the end of the drift tube at different times. Ionic flux at the ion detector will exhibit peaks relating to individual ion species.*

the column coating, causing sensor drift. Cleaning of the input is possible, but can reduce sensitivity.

Different detectors can be used to detect different properties of the gas exiting the column, though they are typically broadly sensitive. It is also common to use a Mass Spectrometer as a detector. In this thesis, an Ion Mobility Spectrometer is used.

Ion Mobility Spectrometry

A number of techniques fall under the category of Ion Mobility Spectrometry (IMS); here we focus on drift tube IMS [see e.g., Eiceman et al., 2013] as used in the FlavourSpec [G.A.S., 2017]. In drift tube IMS, a gas is ionised and the ions are driven down a tube (the "drift tube") by a uniform electric field towards an ion detector. Importantly, this is done in the presence of an inert *drift gas*. This is a buffer gas and travels in the opposite direction to the ion flow. In a vacuum, an ion within an electrical field would accelerate linearly. The presence of the drift gas means the ions quickly reach a terminal velocity due to collisions, the rate of which depends on the mass and geometric structure of the ion. A homogeneous mixture of ions thus separates whilst traversing the drift tube. Ionic flux is measured at the end of the drift tube, so a 1d spectrum of ion flux against drift time is produced.

IMS can produce a spectrum over a short period (tens of milliseconds). This is fast enough to be used as a detector unit following separation in a GC column which operates over much longer time-scales (on the order of 10 minutes for the Flavourspec used in this thesis).

2.3 Analyses and Publications

The following analyses are presented in chronological order. All analyses investigate data produced by one of the three artificial olfaction instruments described in **Section 2.2**. Some key characteristics of each are given in **Table 2.2**.

The first analysis on Whiskey was performed as an exercise to build familiarity with the experimental and data analysis procedures, so all lab work and data analysis was performed by the author. All subsequent analyses are performed on medical data sets collected by collaborators, and the author performed only the data analysis.

The general form of each analysis is that samples are collected from two or more classes (e.g., healthy samples and disease-positive samples), and we show that one is able to construct a statistical pipeline that is able to predict which class a new sample belongs to. In the medical studies, the measurements are from either the breath of patients, or the headspace above a urine sample. Studies showing that the artificial olfaction method is able to distinguish between the classes of interest have been published, which the subsection titles reflect.

Many of these are small sample pilot studies, meaning that we have few samples and thus expect low statistical power. As a consequence we expect, before seeing any data, to be unable to draw any strong conclusions confidently. However, these small sample analyses are still valuable in identifying which future analyses are likely to be the most fruitful.

2.3.1 Scotch Whiskey

To gain intuition for the data collection process, a study was performed in which different scotch whiskeys were run through the FAIMS, generating a dataset of whiskey odours on which a basic analysis was performed. Five different brands of

Study	Technique	Sample type	Section	Publication
Whiskey	FAIMS	Sample headspace	**2.3.1**	
Irritable Bowel Disease	FAIMS	Breath	**2.3.2**	[Arasaradnam et al., 2016b]
Tuberculosis	uvFAIMS	Breath	**2.3.3**	[Sahota et al., 2016]
Hepatic Encephalopathy	uvFAIMS	Breath	**2.3.4**	[Arasaradnam et al., 2016a]
Diabetes	E-nose	Urine headspace	**2.3.5**	[Esfahani et al., 2015]
Bacterial Vaginosis	E-nose	Vaginal swab	**2.3.6**	
Diabetes/Obesity	GC-IMS	Urine headspace	**2.3.7**	

Table 2.2: *The studies performed used a range of artificial olfaction instruments, and used different biological samples to measure. Studies in which sample separation was successful were published.*

scotch whiskey were used (see **Table 2.3**). Four of the bottles are single malts; i.e., produced by a single distillery. A blended whiskey was also included. For each single malt, four 5cl bottles were purchased to investigate within-distillery variation. The FAIMS used was the Lonestar (Owlstone, Cambridge, UK) [Owlstone Nanotech, 2017a,b, 2006; Wilks et al., 2012; Shvartsburg et al., 2009].

The Ardbeg whiskey is "peated". This means that peat smoke is used in production of the whiskey, and gives a strong recognisable smell in the final product. This was included to investigate if the FAIMS can distinguish peated from un-peated whiskeys. A range of maturation times were also selected, as maturation may also be a property the FAIMS can detect.

Data Collection

The FAIMS detector chip is highly sensitive to alcohol; the high alcohol content of the neat whiskey would saturate the response from the sensor and would require decontaminating the sensor for long periods of time (days or more). The whiskey must thus be diluted. 10%, 4% and 2% dilutions were attempted, and 4% was determined to give a good signal strength without saturating the sensor, so was used in the following experiment.

For each of the 4 single malts, samples were taken from 4 different bottles. Only a single bottle of the blended whiskey was used. For each sample, three sequential runs were done, recording three data points. The total number of smell signatures recorded was thus $4 \times 4 \times 3 + 3 = 51$. The methodology used in data collection was as follows:

1. Prepare Sample: pipette 4800ml water and 200ml spirit into sample container.
2. Replace sample in FAIMS with newly prepared sample.

Distillery	Single Malt	Age	ABV	Peated
Aberlour	Yes	10	40%	No
Ardbeg	Yes	10	46%	Yes
Glenfiddich	Yes	15	40%	No
Glenlivet	Yes	18	43%	No
Co-operative Blended Scotch Whiskey	No	–	40%	–

Table 2.3: *Whiskey brands analysed with the FAIMS. A range of maturation ages was selected, as was the presence of a peated whiskey and a blend. The omitted information for the blended whiskey is not commercially available.*

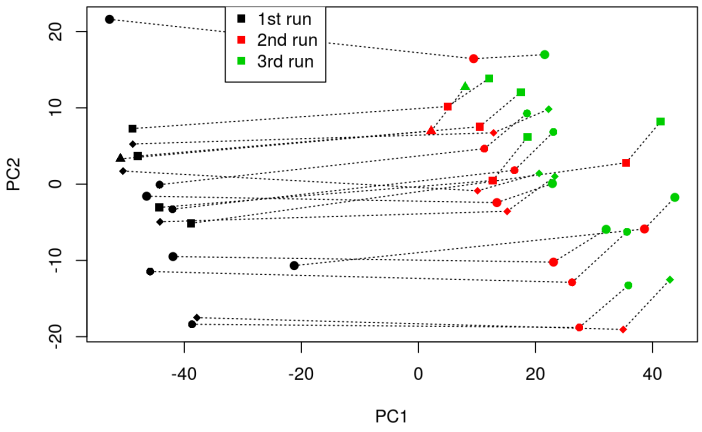


Figure 2.10: *Whiskey data projected onto the first two Principal Components. Each sample was left in the FAIMS long enough to collect three measurements, and each of the three runs of each sample is connected by a dotted line. It can be seen that the first principal component is simply how long the sample has been in the FAIMS. Each type of whiskey has been given a different symbol, and no separation between whiskeys can be seen.*

3. Warm sample for 5 minutes.
4. Start the FAIMS, collecting three data points over 7 minutes 30 seconds.
5. Stop data collection, wait for pressure and temperature to stabilise ($\sim 0.5\frac{\mathrm{L}}{\mathrm{min}}$, ~ 0.1bar).
6. Remove sample, replace with empty sample container.
7. Perform blank runs until the sensor is no longer contaminated.
8. Repeat.

Analysis

Performing PCA on the whiskey data and projecting the data onto the first two principal components, we obtain **Figure 2.10**. Each sample from a single bottle gives three data points due to being left in the FAIMS for thee runs to investigate by how much the runs differ. This turns out to be the largest source of variance in the data, and is thus the first Principal Component of the data. The third run is used for the remainder of the analysis, so each individual bottle corresponds to exactly one data point. This was chosen as the third run has the greatest mean pixel intensity, corresponding to a greater total number of volatiles being measured.

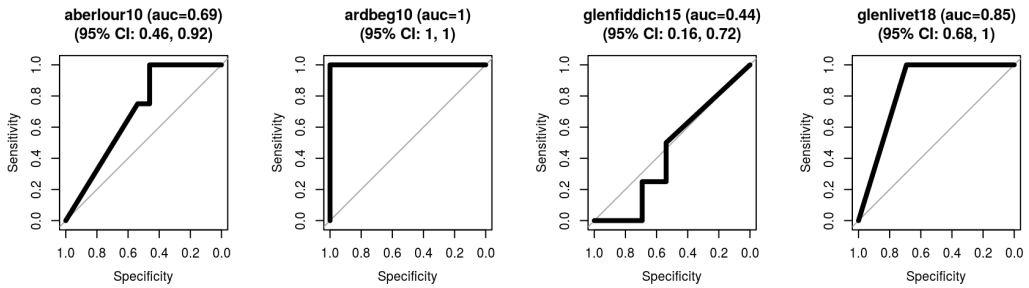


Figure 2.11: *AUCs obtained with Random Forest classifier after supervised dimension reduction. The 4 classification tasks are 1-against-rest for each of the 4 single malt whiskeys. It is clearly an easier task to distinguish the Ardbeg from the other whiskeys, which may be related to it being the only peated whiskey.*

We attempt 4 binary classification tasks: for each sample we classify whether it is from a single malt of interest, or from the remaining 3 single malts. This is done with a leave-one-out cross-validation to produce a set of predictive probabilities, and uses the previously developed pipeline detailed in **Section 2.2.2**. AUCs for each task are displayed in **Figure 2.11**.

Observing **Figure 2.11**, it can be seen that Ardbeg is readily distinguished from the other whiskeys. This may be due to it being peated, which would be an interesting hypothesis to study further. The remaining three single malts do not separate so easily. Glenlivet looks like it separates well and the 95% confidence interval of the AUC does not include 0.5, but more data would be needed to draw the conclusion that the FAIMS can distinguish it easily.

Linear Discriminant Analysis

Linear Discriminant Analysis (LDA) is performed to linearly project the data onto the two dimensions in which each of the 5 classes of whiskey are maximally separated (**Figure 2.12**). Since the number of data points and classes are low compared to the dimensionality of the data, it is unsurprising that the classes can be separated well. To test generalisation of separation across types of whiskey, each whiskey is removed in turn when learning the projection. When applying the projection to the data, the held out data continues to form clusters distinct from the whiskeys used to learn the projection. This shows that, under this statistical model, the FAIMS sees samples from our unseen whiskeys as similar to each other and different to other whiskeys seen.

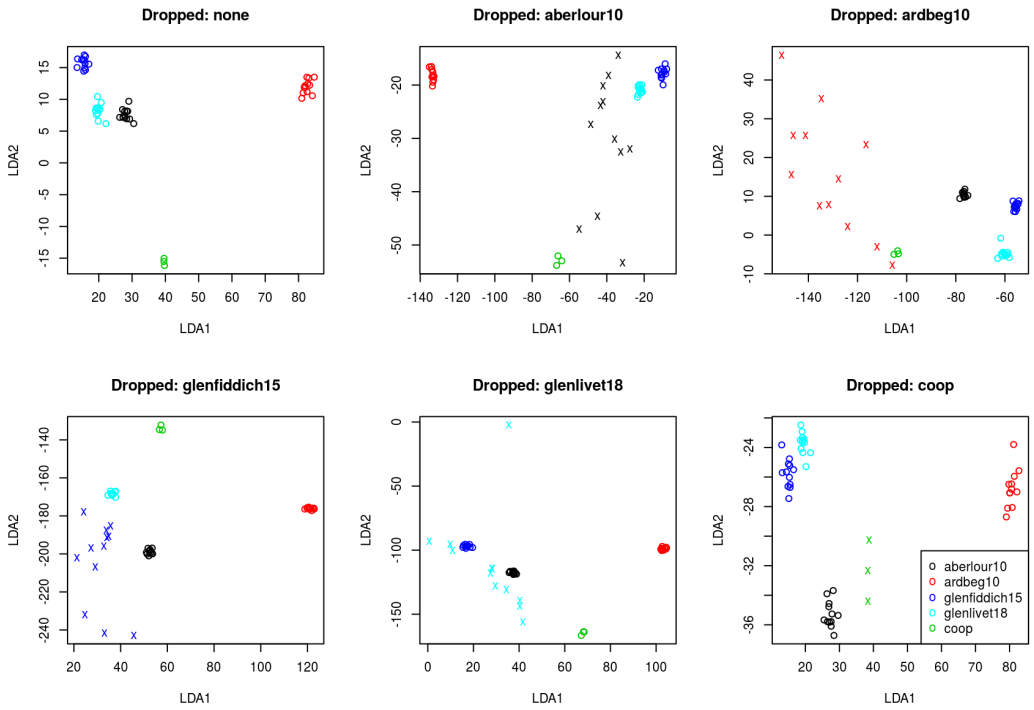


Figure 2.12: *LDA is applied to the whiskey data, reducing the data to two dimensions. Plotting this (top left) shows good separation between groups. To see how the projection generalises to new whiskeys, we learn the projection 5 more times (remaining plots), each with one of the whiskeys omitted during learning. Applying the learned projection to all whiskeys, we see the omitted whiskey is still separated from the others, showing good generalisation.*

Conclusions

This small sample study provides evidence that the FAIMS is, with the correct signal processing pipeline, able to distinguish between certain types of whiskey. With more data perhaps the whiskeys could be distinguished with greater accuracy. The easiest whiskey to distinguish from all others in the study was the only peated whiskey investigated. Further investigation would be required to show that the FAIMS can distinguish peated from un-peated whiskeys, but this study provides support for such a hypothesis.

Care must be taken in the experimental process to produce high quality data. The FAIMS must be decontaminated for sufficient time between running samples, particularly if they contain a compound to which the sensor is particularly sensitive, such as alcohol. The data produced are also sensitive to how long the sample has been in the FAIMS chamber. If a machine learning classifier has been trained on just the third run of each training sample, then only the third run of a sample should

be classified.

The study was only of small sample size, and a number of analyses were attempted, such as including and excluding the cooperative blended whiskey. This kind of high-flexibility and low-power study can often appear to produce significant results when there are none [Simmons et al., 2011]. Whilst it may be sensible to believe the FAIMS is able to distinguish Ardbeg from the other un-peated whiskeys, the 100% accuracy achieved here is unlikely to hold on a larger study.

2.3.2 Non-invasive exhaled volatile organic biomarker analysis to detect Inflammatory Bowel Disease [Arasaradnam et al., 2016b]

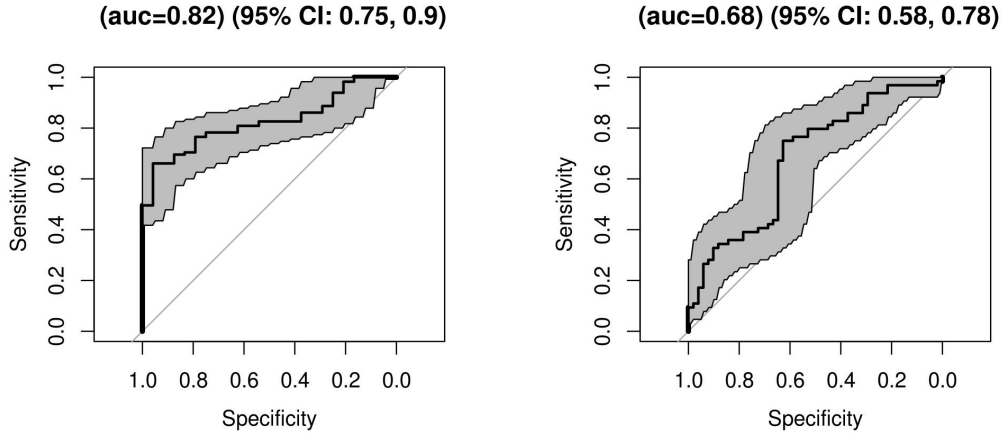
Inflammatory Bowel Disease (IBD) is a class of diseases characterised by inflammation of the bowel and small intestine, causing pain and reducing quality of life [NICE, 2015]. IBD has two main forms: Crohn's Disease (CD) and Ulcerative Colitis (UC). CD and UC affect approximately 115,000 [NICE, 2012] and 146,000 [NICE, 2013] individuals in the UK respectively.

A number of methods exist for diagnosis of IBD, but distinguishing between UC and CD can be difficult when the disease is limited to the colon [Arasaradnam et al., 2016b]. This study was aimed to determine whether the electronic nose can distinguish between breath from IBD patients and healthy controls, and also whether UC can be distinguished from CD.

53 IBD patients (29 UC, 24 CD) were recruited sequentially from dedicated IBD clinics at University Hospitals Coventry & Warwickshire. 11 healthy controls volunteers (V) were recruited such that the controls reported no overt gastrointestinal symptoms and were not on routine oral medication. For each of the patients and controls, morning breath samples were taken following a 2 hour fast. Breath was captured using a custom device in which breath is captured in a Tedlar® bag. Environmental conditions are accounted for by first having the patient inhale filtered air. Captured samples were frozen at −20°C, transported to the University of Warwick and tested using the Lonestar FAIMS unit on the same day as capture.

Classification task	UC & CD vs V	UC vs V	CD vs V	UC vs CD
AUC	0.82	0.75	0.77	0.68
(95% CI)	(0.74–0.89)	(0.65–0.86)	(0.67–0.88)	(0.58–0.78)

Table 2.4: *AUCs and 95% confidence intervals achieved on classification tasks involving breath from control volunteers (V), patients with Crohn's Disease (CD), and patients with Ulcerative Colitis (UC).*



(a) *IBD vs control* **(b)** *UC vs CD*

Figure 2.13: *ROC curves for medically relevant IBD prediction tasks*

The collected data were processed using the previously developed statistical pipeline described in **Section 2.2.2**, using sparse logistic regression as the classifier. Four classification tasks were attempted: UC & CD vs V, UC vs V, CD vs V, and UC vs CD. AUCs are presented in **Table 2.4**. The two groups can be distinguished between in all cases, and discriminating between IBD and healthy control breath has the greatest accuracy. For the interested reader, ROC curves for all tasks have been included in **Appendix B.1.1**.

The experimental procedure collected 2 consecutive FAIMS measurements (which we call 'runs') of each sample. The analysis was performed once with each run, and run 2 produced superior results on all 4 classification tasks, so the results from the second run are presented here. Results from the first run are given in **Appendix B.1.2**. There was also 13.6% more signal in the second run as measured by mean pixel intensity, which possibly reflects greater VOC content explaining the better performance.

The most clinically interesting cases are diagnosis of IBD (UC & CD vs V) and distinguishing UC from CD. ROC curves for these are given in **Figure 2.13**. Whilst our ability to distinguish UC from CD is lower than that of colonoscopy, it is greater than that on-the-spot diagnosis [Arasaradnam et al., 2016b]. Our results here are supported by van Gaal et al. [2017], who show paediatric stool FAIMS measurements can distinguish healthy controls from IBD with AUC of 0.76 (95%CI 0.62–0.9), and UC from CD with AUC 0.9 (95%CI 0.8–1).

The task of discriminating IBD from healthy control samples, and the task of

discriminating UC from CD have some notable differences. Being able to distinguish between IBD and control samples does not imply that FAIMS could be used as a diagnostic test; it could be the case that there is a very general immune response in the body to being unwell, and we could be picking up on that instead of the specifics of IBD. If this was the case, we would only be able to discriminate between well and unwell patients. The fact that we have good separation between the UC and CD samples provides evidence against this; UC and CD are very similar diseases, both causing inflammation of the gastrointestinal tract, yet we are able to discriminate between the two. This shows that FAIMS can identify disease-specific signal.

The result of discriminating UC from CD is more likely to hold up in a real world medical application than the result for diagnosing IBD. The reason is that the population chosen for this study was patients attending an IBD outpatient clinic, whereas attempting to diagnose IBD in the general population is a more difficult task due to increased diversity in the population.

It is interesting that IBD, a condition occurring in the gastrointestinal tract, can be diagnosed and categorised using breath measurements. Previous work relating to this includes FAIMS and E-nose diagnosis of IBD [Arasaradnam et al., 2013], as well as discriminating between those with active IBD and those in remission.

2.3.3 A simple breath test for tuberculosis using ion mobility: A pilot study [Sahota et al., 2016]

Tuberculosis (TB) is an infectious bacterial disease. In 2015 there were 10.4 million new TB cases worldwide, and 1.4 million TB deaths [World Health Organization, 2016]. This study [Sahota et al., 2016] investigates the use of FAIMS in distinguishing patients with TB against healthy control volunteers (V) using patient breath. This would be an attractive mechanism for diagnosis, since the standard method of microbiological culture can take up to 2 months [Sahota et al., 2016].

This study uses ultra-violet (UV) light as the ionization source instead of the radioactive isotope Ni63. UV does not ionize the same range of chemicals as Ni63, so can produce a lower quality signal. However, avoiding radioactive isotopes means the FAIMS can be transported easily and used in hospitals without special licensing, allowing the samples to be analysed more rapidly and thus undergoing less degradation.

Over 6 months, 21 adults with TB were recruited from the University Hospitals Coventry & Warwickshire with suspected TB, before or within 1 week of starting anti-TB medication. 19 healthy controls were also recruited.

Breath was captured by each participant breathing through a mouthpiece

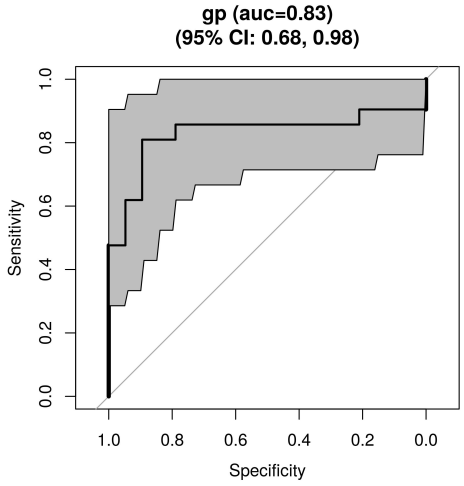


Figure 2.14: *ROC curve for classification pipeline on TB vs Controls classification task using Gaussian Process (GP) classifier.*

with a two-way valve until a 3L Tedlar® bag was filled. Samples were then analysed within 2 hours of collection by the Lonestar uvFAIMS (Owlstone, UK).

Using the previously developed statistical pipeline (**Section 2.2.2**) with leave-one-out CV, each sample was classified as healthy or TB using a Gaussian Process classifier[5]. We obtain an AUC of 0.83 (95%CI: 0.68 – 0.98) (**Figure 2.14**). We use the second of 2 available runs due to greater mean signal.

Two patients in the control group were initially suspected of TB but were finally given alternative diagnoses of non-TB bacterial infections. These are of interest to study; perhaps their similar disease will confuse the classifier into labelling them as TB positive. This does appear to be the case, since their assigned predictive probabilities of belonging to the TB class are greater than is characteristic for points from the Control class (**Figure 2.15**).

The results here differ slightly to those described in the publication relating to this work Sahota et al. [2016] since the analysis has been re-run with leave-one-out cross-validation (instead of 10-fold), and the Gaussian process classifier has been switched to the one written by the author.

Our results agree with previous work, obtaining similar levels of predictive accuracy. Breath has been identified as a possible route for TB diagnosis using GC-MS [Kolk et al., 2012] and E-nose [Bruins et al., 2013]. Sputum has also been measured with E-nose [Kolk et al., 2010], with slightly lower accuracy.

[5]Implemented by the author as described in [Rasmussen and Williams, 2005]. R package hosted at `https://github.com/JimSkinner/gpclassifier`

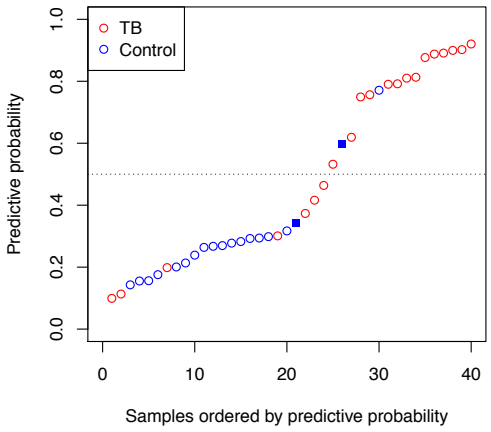
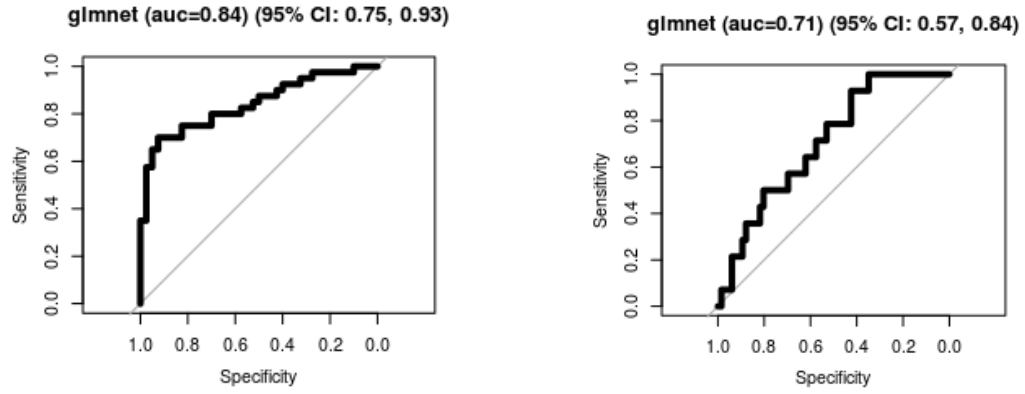


Figure 2.15: *The predictive probabilities of each point having TB. Points have been sorted by their assigned predictive probabilities. Colours indicate the true class of each point, and a good separation between classes can be seen. The two filled squares indicate points in the control class that were initially misdiagnosed as having TB, and it can be seen that these have been more difficult to classify.*

2.3.4 Breathomics—exhaled volatile organic compound analysis to detect hepatic encephalopathy: a pilot study [Arasaradnam et al., 2016a]

Hepatic Encephalopathy (HE) is a disturbance in brain function resulting from liver disease. Symptoms range from minor changes in consciousness, memory and concentration to confusion, amnesia and coma [Cash et al., 2010]. HE is categorised into covert and overt HE; symptoms of covert HE are absent or minor, whilst overt HE shows much greater mental disturbances. Whilst there are treatments available for both covert and overt HE, these treatments are typically given only for overt HE due to ease of diagnosis [Arasaradnam et al., 2016a]. It is thus a clinically useful task to detect covert HE.

13 patients with covert HE, 9 patients with overt HE, and 20 control volunteers were recruited. The control group comprised of healthy volunteers without clinical or biomedical evidence of liver disease. Breath samples were taken following a 2 hour fast and absence of cigarette smoking. A custom breath capture device was used, allowing natural breathing through a mouthpiece until a 3L Tedlar® bag is filled. Only end-tidal breath is collected, which has a higher VOC content. Samples were then frozen at -20°C and transported to the University of Warwick, where the uvFAIMS was used to collect smell measurements.



(a) *Classifying HE from controls.* **(b)** *Classifying covert from overt HE.*

Figure 2.16: *HE classification task ROC curves*

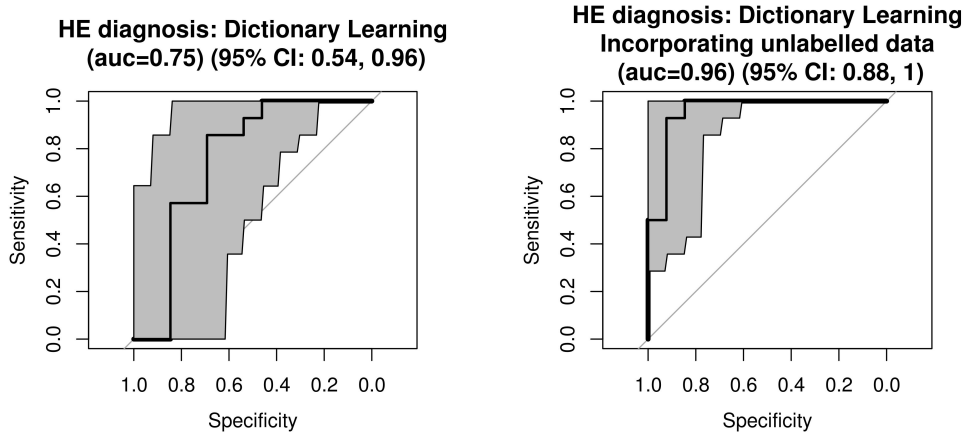
The previously mentioned statistical pipeline (**Section 2.2.2**) was used to classify samples. Two classification tasks were decided upon: classifying HE (covert or overt) from controls, and classifying covert HE from overt HE. On HE classification we obtained an AUC of 0.84 (95%CI 0.75–0.93), and on distinguishing covert from overt HE we obtained an AUC of 0.71 (95%CI 0.57–0.84). The relevant ROC curves are given in **Figure 2.16**.

Semi-supervised learning

This is a small sample study (n=42), meaning that the feature learning stage may not produce features that are optimum for good classification. Following publication of [Arasaradnam et al., 2016a], we investigated including additional unlabelled data in this feature learning stage to see if this could improve predictive accuracy.

We use the previously developed pipeline (**Section 2.2.2**) but use Dictionary Learning (DL) as a replacement for the wavelet transformation as the feature learning stage. In addition, the classifier and the number of retained features in the supervised Wilcoxon rank sum feature selection stage are chosen on a training set comprising two thirds of the data. The remaining third of the data is then used to produce a ROC curve.

This procedure was repeated twice with one variation: in one run the DL stage operates only on the data at hand; in another run DL is performed on a larger dataset of 141 samples made from combining a number of previous studies in addition to the HE data. The AUCs on the test set for each variant were 0.75 (95%CI 0.54–0.96) and 0.96 (95%CI 0.88–0.1) respectively; ROC curves are given



(a) *AUC on classifying HE from healthy controls using standard pipeline with Dictionary Learning as a replacement for the fixed wavelet transformation.*

(b) *AUC obtained with the same pipeline as* ***Figure 2.17a****, but including additional unlabelled FAIMS data in the Dictionary Learning stage.*

Figure 2.17: *Predictive performance appears to increase when including additional unlabelled data in the feature learning stage.*

in **Figure 2.17**. This shows evidence that augmenting the feature learning stage with a corpus of unlabelled data has enabled better predictions. The small size of the validation set means the uncertainty around the AUCs are large; a larger study be better able to show whether or not a true improvement was made.

2.3.5 A rapid discrimination of diabetic patients from volunteers using urinary volatile and an electronic nose [Esfahani et al., 2015]

This subsection details a study into the ability of the FOX4000 electronic nose to distinguish between urine from type-2 diabetic patients and healthy controls. This is an interesting case study as the results initially appeared good, with the statistical pipeline obtaining 100% accuracy on predicting whether samples in a validation dataset were from control or diabetic patients, leading to a published conference abstract and talk [Esfahani et al., 2015]. However, further scrutiny revealed that this predictive accuracy was due to a flaw in the study design.

91 urine samples taken from 43 type-2 diabetic patients and 48 control volunteers were collected at University Hospital Coventry & Warwickshire. Samples were collected in clinic and frozen to -80°C within 2 hours. For E-nose analysis, urine samples were thawed to 4°C for 24 hours, and 3ml of sample was aliquoted

into 10ml glass vials.

The dataset of 91 samples was split into a training dataset of size 61, and a validation dataset of size 30. The split was done by ordering the samples by the run date and placing every third sample in the validation set. The training dataset was used to construct the statistical pipeline, and this was then applied to the validation dataset to estimate the predictive accuracy.

The pipeline to produce classifications was chosen to be a feature learning stage followed by a classifier. Using the training set, ICA (Independent Component Analysis, **Section 1.1.1**) was selected as the feature learning stage, and the Random Forest as the binary classifier.

Model Selection

A pipeline is evaluated on the training dataset by reducing the dimension of the data with a feature learning technique, then producing predictive probabilities with a classifier inside of a 10-fold cross-validation. The predictive probabilities produced were compared to the known labels to produce an AUC; the higher the AUC, the better the pipeline. A number of feature learning techniques and classifiers were applied to the training set, and the best-performing pair in terms of training AUC were then applied to the validation set.

The four classifiers investigated were the Random Forest [Breiman, 2001], SVM [e.g., Bishop, 2006], Sparse Logistic Regression [e.g., Hastie et al., 2015] and Gaussian Process Classifier [Rasmussen and Williams, 2005]. ICA, PCA, the 1D wavelet transformation, and a collection of hand-engineered features were all considered at the dimension reduction stage. These are each described below, and the training AUCs produced are given in **Table 2.5**.

The hand-engineered features were suggested by Dr James Covington, having been previously used in processing of E-nose data with the MultiSens Analyser software package [JLS Innovation, 2009, Section 4.1]. Each of the 18 sensors is reduced to 2 features each, giving 36 dimensional data. The first feature is the difference between the maximum and minimum value achieved by the curve, which captures how much a sensor responds to the gas presented. The second feature is the area under the sensor response curve after having the sensor value at time 0 subtracted from the entire curve. Alone, this feature cannot distinguish between sensor responses which are intense and brief, and those which do not reach a high maximum but take a long time to return to the baseline after the gas is removed. However, when paired with the first feature this allows the classifier to use information on how quickly a sensor returns to baseline.

Dimension Reduction	Feature Count	Training AUC			
		Sparse Linear Regression	**Random Forest**	**SVM**	**Gaussian Process Classifier**
Hand-engineered features	36	0.671	0.922	0.923	**0.937**
Wavelets	2	0.856	0.923	0.945	0.883
	3	0.852	0.927	**0.952**	0.895
	4	0.856	0.928	0.949	0.893
	5	0.856	0.922	0.942	0.923
	6	0.853	0.918	0.938	0.917
	7	0.850	0.918	0.938	0.933
	8	0.859	0.928	0.938	0.935
PCA	2	0.463	0.906	0.881	0.903
	3	0.463	0.923	0.900	0.935
	4	0.714	0.994	0.991	**1.000**
	5	0.948	0.997	0.971	0.991
	6	0.954	0.999	**1.000**	**1.000**
	7	0.943	**1.000**	**1.000**	**1.000**
	8	0.957	**1.000**	**1.000**	**1.000**
ICA	2	0.965	0.856	0.901	0.899
	3	0.588	0.967	0.858	0.942
	4	0.703	**1.000**	0.990	0.996
	5	0.942	0.993	0.974	**1.000**
	6	0.934	**1.000**	0.998	0.997
	7	0.964	0.996	**1.000**	**1.000**
	8	0.970	**1.000**	**1.000**	1.000

Table 2.5: *AUCs obtained on the training data in a 10-fold cross-validation for every dimension reduction technique and classifier attempted. We see that PCA and ICA perform similarly, and are the only techniques to obtain an AUC of 1.*

The wavelet decomposition is applied to each sample separately by concatenating all sensor traces into a single vector and applying the 1d wavelet decomposition using the `wavethresh` R package [Nason, 2016]. This produces 4095 sparse wavelet coefficients, of which only 2496 have non-zero values over the entire training set so are retained. This is still a large number of features, so in the cross-validation stage we include an additional supervised feature selection stage, using only the in-fold training data to learn which features to select and applying this to the test data. The Wilcoxon rank-sum test is performed on each of the 2496 features, testing the

null hypothesis that the feature distribution for the disease and control groups is the same. This produces a p-value for each feature, and we retain only the k features with the lowest p-values. k is varied from 2–8, producing the AUCs in **Table 2.5**. Comparing the wavelet and hand-engineered feature AUCS, we see the wavelets out-perform the hand-engineered features.

The wavelet decomposition was considered due to its good performance on the FAIMS data in the previously developed pipeline (**Section 2.2.2**). Most features produced by the wavelet transformation use information only from a single sensor, since each wavelet feature captures signal around a particular location. The only features that capture signal across sensors will be very wide wavelets (since the response from all sensors are concatenated together), or wavelets at the boundary between sensors. This differs from PCA and ICA in which every feature uses information from every sensor.

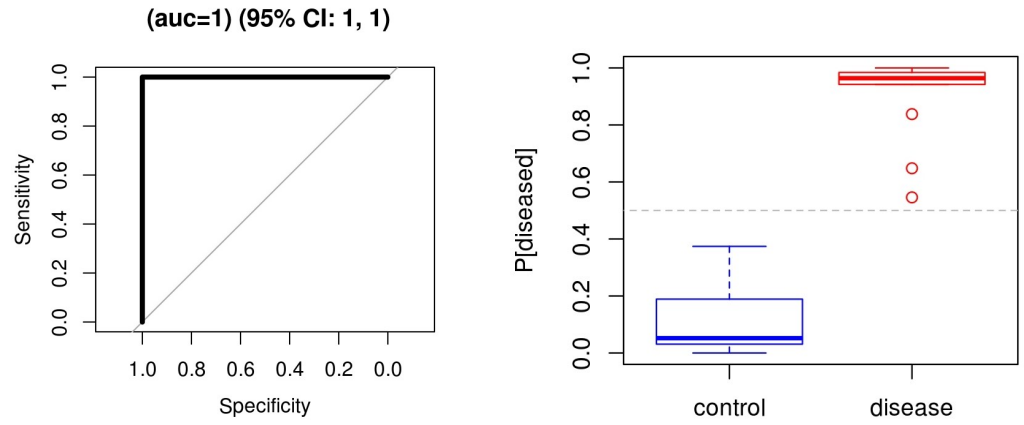
PCA and ICA were applied for a range of latent dimensionalities. Due to the finite size of our dataset, a number of models achieved perfect classification producing an AUC of exactly 1. There is thus insufficient information to choose the 'best' model, so we invoke Occam's razor and select the simplest model achieving perfect classification. With a latent dimensionality of 4, both PCA and ICA are able to achieve an AUC of 1. We decided upon the model using an ICA reduction to 4 dimensions followed by classification by a Random Forest classifier. However, an equally valid choice would have been PCA to 4 dimensions followed by classification by a Gaussian Process Classifier.

Results on validation dataset

By selecting the model maximising cross-validated AUC, it is expected that we have over-fit and the AUC of 1 achieved is optimistic. To estimate the generalisation performance we apply the selected pipeline to the validation data.

Predictive probabilities are produced for the validation data using the same pipeline, and including the training data in the feature learning and classifier training. We again obtain an AUC of 1 (**Figure 2.18a**), perfectly classifying every validation sample and indicating a high predictive accuracy.

Most of the predictions produced are highly confident since the predictive probabilities are close to 0 or 1. **Figure 2.18b** shows that a range of classification thresholds of approximately 0.4–0.55 would give perfect classification.



(a) *ROC curve for predictions produced on validation dataset. We obtain an AUC of 1, since all samples may be classified perfectly.*

(b) *The predictive probability distributions for the control and disease classes set are non-overlapping. Selecting a classification threshold within the approximate region of 0.4–0.55 would give perfect classification of the validation data.*

Figure 2.18: *Predictions on the validation data are accurate and confident.*

Investigating the feature space

Investigating the 4 dimensional feature space is interesting. By considering just two of the dimensions, the data can be plotted and coloured by its disease class for visual inspection. This is given for all pairs of features in **Figure 2.19**. The first feature (V1) is good at separating the classes, particularly when combined with the 4th feature (V4).

Combining features V1 and V4 appears, from **Figure 2.19**, to be sufficient for highly accurate classification. This leads to the question of why a latent dimensionality k of 4 and not 2 was selected. The reason is that ICA is unsupervised, and in the $k = 2$ case the feature mappings learned did not correspond to V1 and V4. ICA discovers underlying signals in the data regardless of their relevance for classification, and $k = 4$ was necessary in this case to uncover the informative features V1 and V4. Given a larger dataset it may be useful to incorporate an additional feature selection stage, retaining only the ICA dimensions useful for disease classification.

In the V1/V4 pair useful for classification, the disease signal is clustered and the control signal more spread-out. One can expect this since, for disease-positive samples, the disease signal swamps all other signal, making all disease-positive samples appear similar. The representation of the control samples is thus dominated by non-disease related signal, which displays much more variation.

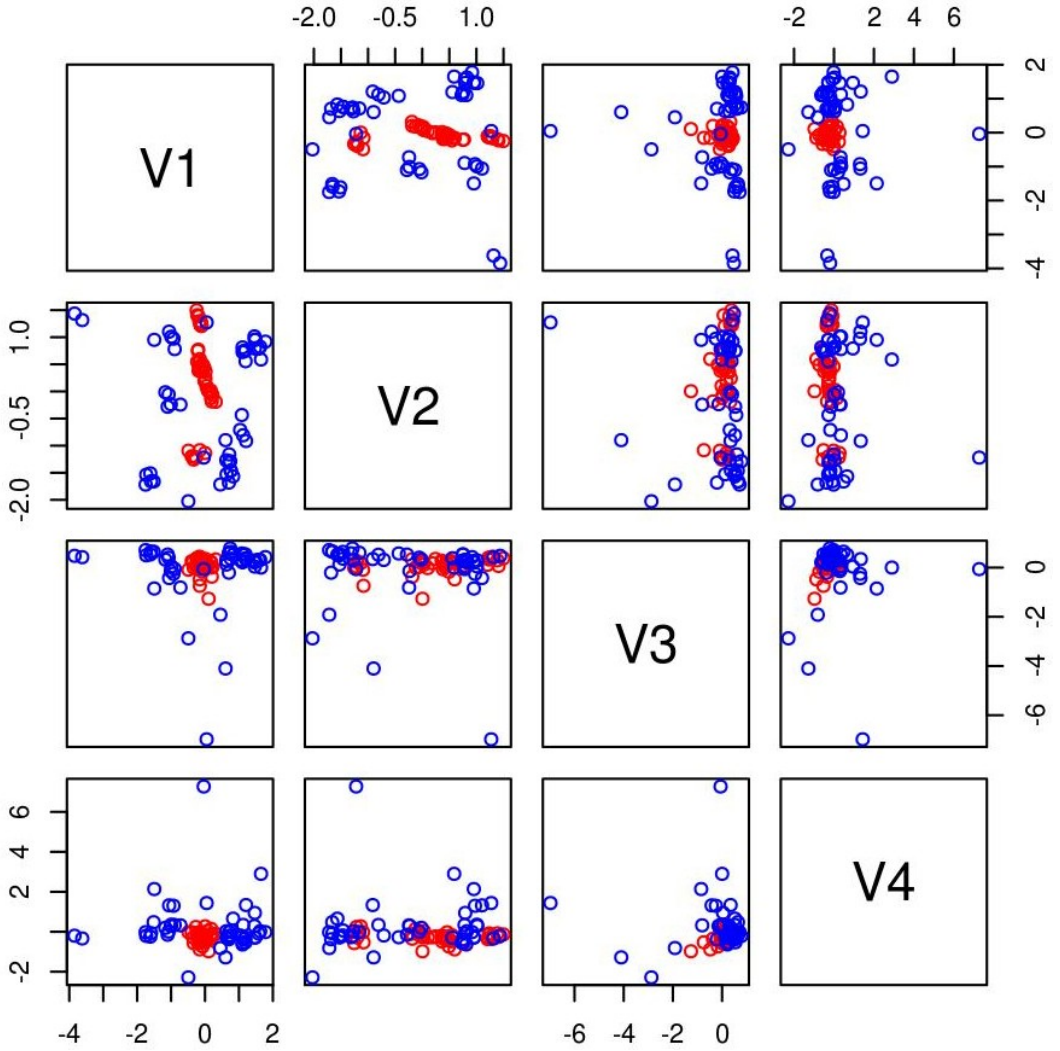


Figure 2.19: *2d projections of the data into pairs of ICA dimensions. Colours indicate diabetic (red) or control (blue). The V1/V4 projection separates the classes particularly well.*

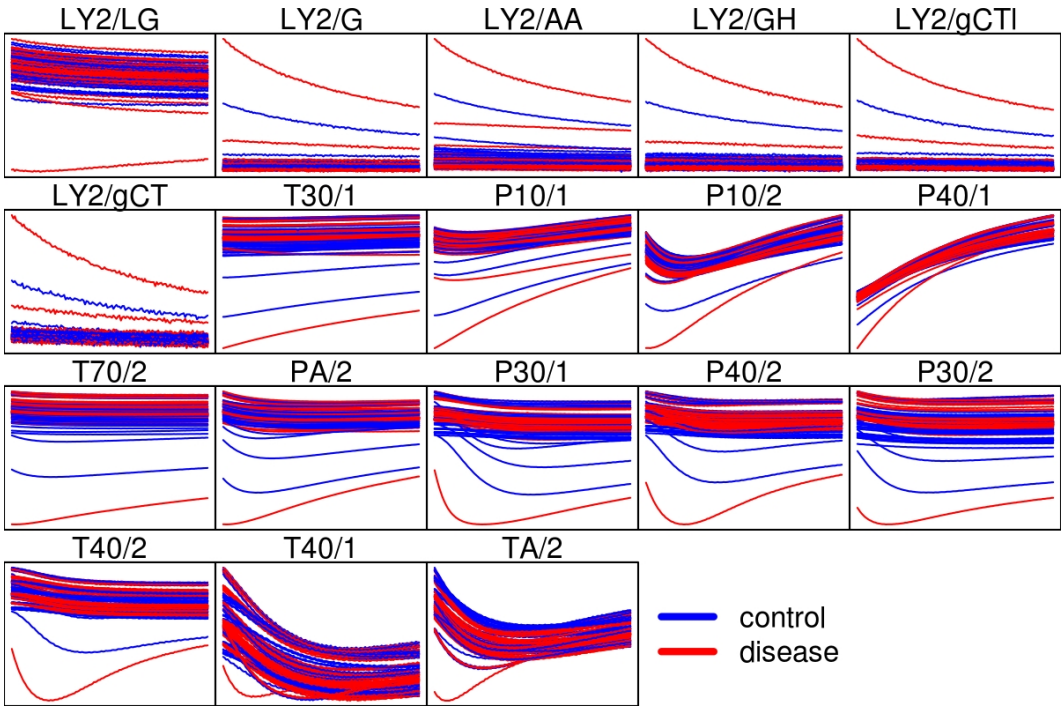


Figure 2.20: *Each box shows the time-series for a single sensor for every sample, coloured to indicate class (red for disease, blue for control). No obvious difference in sensor responses can be seen between classes. However, we know that from these samples we can achieve very good classification. This must mean that the signal is spread across multiple sensors.*

The signal is spread across sensors

Each of the 18 sensors in the FOX4000 produces a time-series as the gas from a single sample is passed through the sensor array. By plotting the time-series of all samples in **Figure 2.20** and colouring by the sample class (red for diabetic, blue for control) we can search visually for differences in sensor activations between classes.

There is no clear visual difference between the red and blue sensor responses. However, we know that the classes can be distinguished between very accurately. Furthermore, the hand-engineered features and the wavelet decomposition both extract features which take information from only a single sensor per feature. These local feature extraction techniques under-perform both PCA and ICA which extract global features, taking information from all sensors. There must thus exist a strong signal that is spread across multiple sensors. This illustrates the use of feature learning techniques for electronic noses; informative features require information from all sensors, but this would be difficult to design by hand.

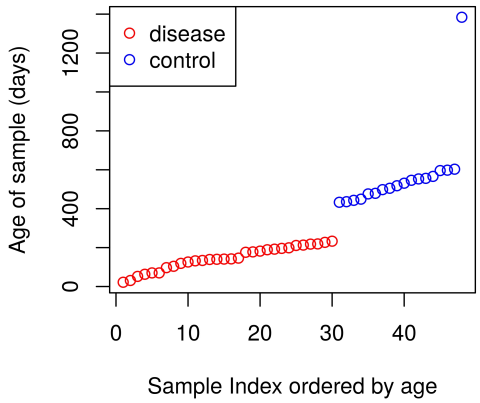


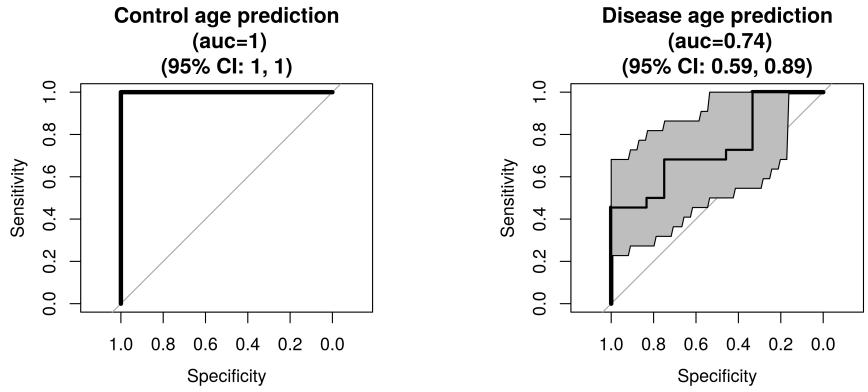
Figure 2.21: *Different samples were stored for different periods of time before being measured by the electronic nose. Ordering samples by their age and colouring by the sample label (diabetic – red, control – blue), it can be seen that all control samples were older than 433 days, and all diabetic samples were younger than 233 days at the time of measurement.*

Where the study failed

Along with the electronic nose measurements, we also have clinical data about each patient and other information about each sample. To further confirm the validity of our result, we check for confounding variables; i.e., whether the presence of diabetes is associated with any demographic information. If this was the case, it may be that it is not the presence of diabetes that we are able to detect with the electronic nose, just some demographic property which, in the dataset we have, happens to associated with diabetes.

To test for such an association, we try to predict the diabetes label from the patient age, date of birth, gender and BMI. We also include the age of the sample; the period of time between when the sample was collected and when it was measured with the electronic nose. Using the Random Forest, SVM, Sparse Logistic Regression and Gaussian Process classifiers, we do cross-validation and obtain an AUC of 1 from every classifier. This indicates that the diabetes label can be predicted completely from this auxiliary information without using the electronic nose data, and therefore our study may be misleading.

Further investigation shows that the control and disease samples were stored frozen at −80°C for different periods of time (**Figure 2.21**). The range of ages for the diabetic samples was 22-233 days, whilst the range of ages for the control samples was 433–1384 days. This is problematic as the VOC content of a sample



(a) *Control age prediction* **(b)** *Diabetic age prediction*

Figure 2.22: *Investigating the information about the sample age contained in electronic nose measurement. Within the training set, a cross-validated model could predict the age of the control samples accurately. Ideally we would be able to predict the sample class but not the sample age.*

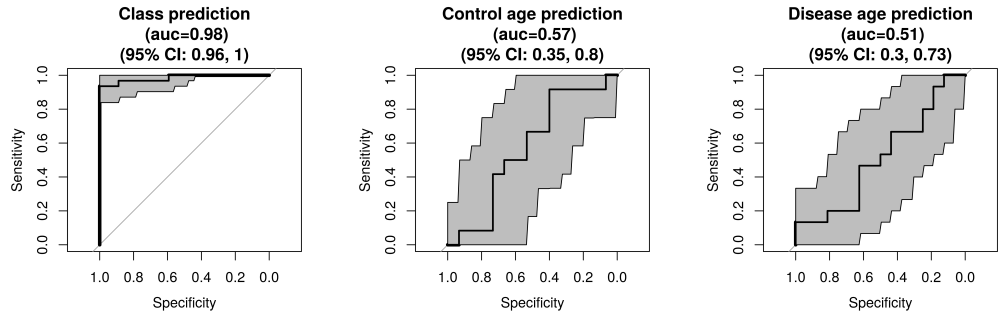
may change over time during storage. It is possible that our model has learnt only to distinguish between samples stored for less than 233 days or greater than 433 days.

We investigated whether the electronic nose measurements contain information about the age of the sample, and thus whether our model could simply have learned to classify the samples by age. To do this we trained a classifier to predict whether the age of each sample is above or below the mean sample age. This was done separately for the control and disease groups. We achieved the ROC curves in **Figure 2.22**, which show that the relative age of a sample in the control group can be predicted with high accuracy, even within each group.

Being able to predict the sample age this way shows there is sample age information in the electronic nose measurements. It is possible that the classifier is using this information to produce accurate classifications. In short, the study data are confounded by sample age, and we cannot say anything meaningful about disease detection.

Sample age compensation

We attempted to remove the sample age signal from the data, and investigate if the disease state can still be predicted after the age signal has been removed. Using only the training set, we fit models of how samples change over time and subtract this from the data. Ideally this would remove all information relevant to the sample age,



(a) *Class prediction ROC* **(b)** *Control age prediction ROC* **(c)** *Diabetic age prediction ROC*

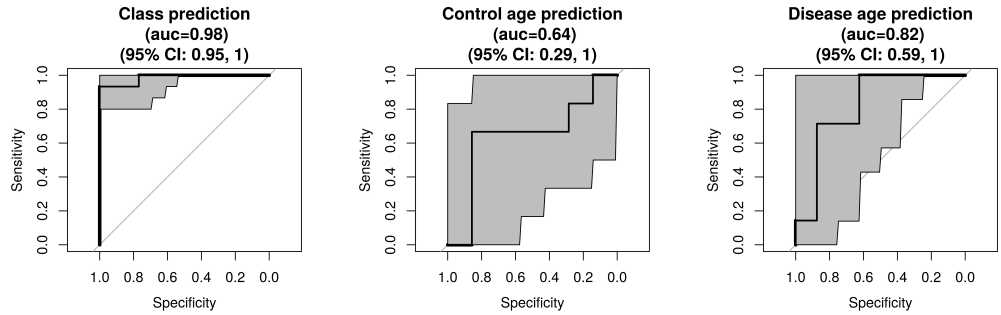
Figure 2.23: *By modelling how a sample changes with age, we can subtract the age related signal from each sample. We try a number of models on the training set to do this, and select that producing the ROCs shown. We are successfully able to remove the ability of the predictive pipeline to predict the sample ages, whilst keeping the ability to predict the sample disease classes.*

making sample age prediction difficult. If there existed any signal in the original data relating to the disease class beyond that of the sample age, the age-adjusted data should still be predictive of the disease class. We would thus ideally achieve poor predictions of the sample age, but good predictions of the sample disease class.

Three classification tasks are considered for the age adjusted data: classification of the disease class, classification of whether the control samples are older than the mean control sample age, and classification of whether the diabetic samples are older than the mean diabetic sample age.

A number of age compensation techniques were attempted. The best achieving method was to first reduce the data to 19 dimensions with ICA. A separate 1d linear regression was fit from the sample age to each of the ICA features. This allowed a point in ICA feature space to be predicted from the sample age alone. The age-predicted ICA representation of each sample was then subtracted from the ICA representation previously obtained from the electronic nose data. Separate regressions were performed for the control and disease samples. Sample disease class was then classified using an SVM with the Gaussian radial basis function kernel.

Using the age-adjusted training data to predict the disease class, control sample age and disease sample age, we obtain AUCs of 0.98 (95%CI: 0.96–1), 0.57 (95%CI: 0.35–0.8) and 0.51 (95%CI: 0.3–0.73) (**Figure 2.23**). This shows good ability to remove age signal and retain disease signal. However, applying the same pipeline to the validation set (including the training data in the feature learning stage), these AUCs become 0.98 (0.95–1), 0.64 (0.29–1) and 0.82 (0.59–1) respect-



(a) *Class prediction ROC* **(b)** *Control age prediction ROC* **(c)** *Diabetic age prediction ROC*

Figure 2.24: *A sample age compensation model is trained on the training set and applied to the validation set. When predicting disease class and sample ages on the adjusted validation set, we find the classifier still has good ability to predict the are of the disease samples (right-hand ROC), meaning the age adjustment does not generalise from the training set.*

ively (**Figure 2.24**).

The predictive performance of disease class between the test and validation data is very similar, showing the age adjustment does not damage the relevant signal. However, it is still possible to predict the age of the diseased samples so the disease signal is still confounded by age. Furthermore there is large uncertainty on the sample age AUCs due to the small sample size. Further investigation thus puts us at risk of discovering a false positive, so we do not attempt further to compensate for age.

Discussion

We developed a model on the training set and achieved 100% predictive accuracy when applying this to a validation dataset. However this was misleading and the model likely has no ability to detect disease, and would not generalise. The reason for this result was that the data for the control and disease classes were stored for different lengths of time, and the amount of time in storage could be detected by the electronic nose.

We attempted to remove the signal related to the sample storage time by fitting a model to how samples degrade over time and subtracting this from the data. However we were unable to find a good model that could generalise well to points not used in model development. Furthermore, even if it did appear that we had removed the sample age signal whilst retaining good disease prediction this would still not give us confidence in the ability for the electronic nose to detect diabetes. The reason for this is that one can never be absolutely certain of having

removed all age related signal from the dataset.

The primary lesson from this analysis is that study design is important. The study was designed and run without consulting a statistician, who would likely have discovered the issue with sample age counfounding at an earlier point. To have confidence in a positive result, the sample disease class (diabetic/healthy) must not be predictable from the sample collection date, storage time or run date, nor from any other demographic information.

The apparently good result was only discovered to be false from actively searching for confounding factors. This kind of rigour is necessary to avoid misleading results such as those described here entering the body of scientific literature.

As illustrated here, using a subset of the data from a single experiment to evaluate generalisation performance can still be overly optimistic. Here the training/validation split was used, but since these were subsets of one dataset generated by a single ongoing experiment, the same experimentally-induced invalidities existed in both splits. Had a separate experiment been performed, the model would have performed poorly on the new data. The validation set would ideally be generated from such a separate experiment, though this may be time consuming and costly.

Useful insights can still be drawn from this study. The difference between the older control samples and the younger diabetic samples was not measured by a single sensor, but was distributed across many, perhaps all. ICA performed well at extracting this information, and this may generalise across datasets.

This dataset was used in Esfahani et al. [2016], who investigate how the FAIMS signal from urine is impacted by the time of the urine in storage, and suggest that urine samples be stored for no more than 12 months, ideally 9, before headspace measurement by FAIMS.

2.3.6 Bacterial Vaginosis

This section describes an exploratory data analysis using FOX4000 E-nose measurements of 66 vaginal swabs. Three diseases were investigated: Bacterial Vaginosis (BV), Group B Streptococcal infection (GBS) and Candidiasis (candida). The primary goal of the analysis was to distinguish between the three diseases, as well as to distinguish between diseased and healthy control samples.

The samples were collected and run in two batches, the details of which are summarised in **Table 2.6**. The batches were used as training and validation datasets, containing 29 and 37 samples respectively. This is a more robust test of the generalisation properties of the classifier compared to a random split; if a model selected on the training set performs well on a validation set generated by a different

Dataset	BV	GBS	Candida	Control	Total	Collected	Run
Training	5	3	6	16	29	10/6/15–22/7/15	24/8/15–26/8/15
Validation	8	5	5	23	37	05/8/15–30/9/15	13/10/15–20/10/15

Table 2.6: *Case counts and collection/run date ranges for training and validation datasets. The "total" column is less than the sum of the "BV", "GBS", "Candida" and "Control" columns as some patients were found to have multiple diseases.*

experiment, then this shows that the model is robust to batch effects, which would be required in a real world application.

The headspace of each swab was measured twice; once dry, and once after the addition of potassium hydroxide (KOH) solution. The standard diagnostic procedure for BV involves identification of a fishy odour after addition of KOH [NGC, 2012], so we may expect a-priori that addition of KOH helps with E-nose BV diagnosis.

There are a number of classification tasks of interest. Accurate classification of the type of infection would be clinically useful. Furthermore, if the precise disease can be identified this shows that the E-nose is able to detect disease specific signal instead of simply being able to distinguish well from unwell patients. Candida is a fungal disease whilst BV and GBS are bacterial, so it is of interest to test if the electronic nose can distinguish fungal from bacterial infections, which require different medical treatments. However, testing a large number of hypotheses on a small dataset is likely to lead to false positives. We thus focus on discriminating between Candida-positive samples and all other samples, since this is the largest class available and being the only fungal disease it may be the easiest to detect.

Data pre-processing

The raw data are plotted in **Figure 2.25**. The very left of each sub-plot shows the sensor response value before any gas is passed over the sensor. This differs between different samples, but ideally would be the same. This additional source of variation is removed by subtracting the starting value from each sensor trace, causing each trace to start at zero. We also then divide each sensor trace by its standard deviation, as we find that this also correlates with the starting value. The sensor traces after this pre-processing stage are shown in **Figure 2.26**.

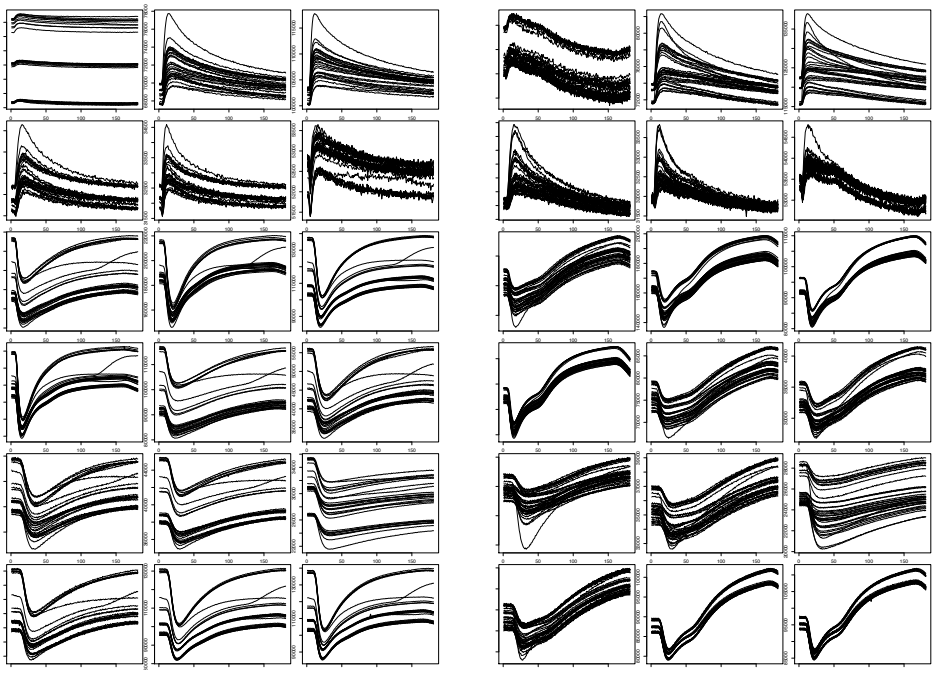


Figure 2.25: *Left: dry. Right: KOH treated samples. Each box is a single sensor, where the sensor response for every sample is shown. Different samples start with different baselines, though this value is reported by the sensor before any sample is seen. The value of the baseline affects whether the sample under or overshoots the baseline at the end of the time-series.*

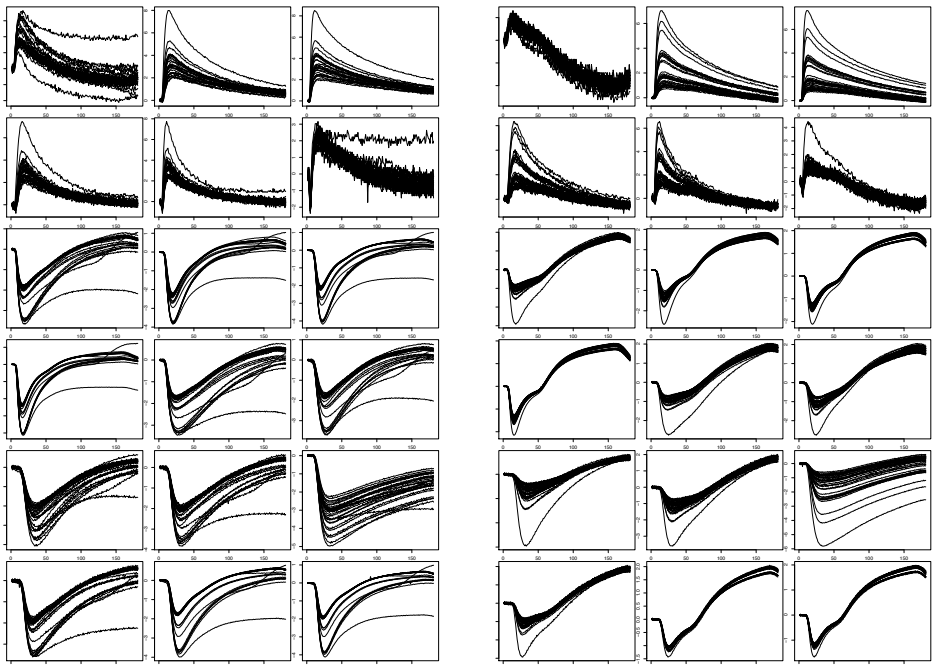


Figure 2.26: *Left: dry. Right: KOH treated samples. Sensors are standardised by subtracting their initial value and scaling all time-series' to have standard deviation 1. Some sensors return to a value other than their original baseline.*

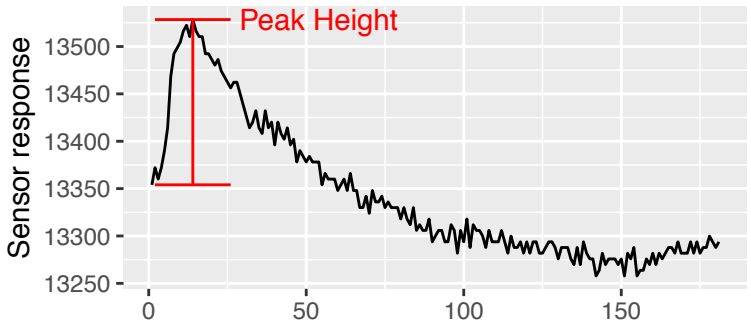


Figure 2.27: *The "Peak Height" model represents an entire sensor trace as the difference between the initial value and the peak height.*

Peak height and sensor level models

The FOX4000 has 18 separate sensors, each responsive to a different property of the gas being measured. In theory, the sensor response sits at some baseline value and, once the gas being analysed starts being passed over the sensor, the sensor will ramp up to some maximum value dependent on the gas. Once the gas stops being passed over the sensor, the sensor response will return to the baseline value. This turns out not to always hold true in practise, as is discovered during the analysis.

The output from the FOX4000 is a time-series of sensor responses for each sensor. Each sensor is sampled 181 times during the run, giving a 181 dimensional measurement from each sensor. However, the sensor response is quite simple and can likely be summarised in a much smaller number of dimensions, producing a low dimension feature representation of the sensor response. We investigated two "sensor level models" in this analysis, which consider each sensor independently, and use prior knowledge of the physics of the sensors to compute a small set of features summarising the response time-series.

The sensor level models here have been constructed specifically for extracting features from an electronic nose sensor response time-series. This contrasts to using flexible and widely applicable feature learning techniques with many parameters such as PCA and ICA. Such techniques do not incorporate our prior knowledge of the sensor physics, but may learn structure that we did not think to look for. These are investigated after the sensor level models.

The simplest sensor level model we refer to as "Peak Height", which represents a sensor trace as the difference between the baseline sensor value and the sensor value at the peak signal. This is illustrated in **Figure 2.27**. The Peak Height model makes the assumption that all of the information in a single sensor trace is

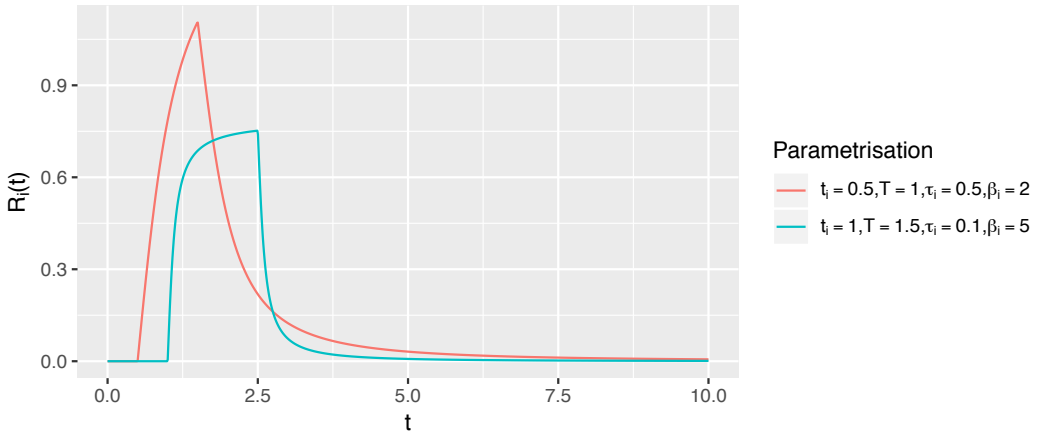


Figure 2.28: *The Lorentzian model for a single sensor as described by* ***Equation 2.2****. The two curves correspond to different parametrisations in the model.*

contained within the height of the peak. All other information, including the rate of change of the signal, the final value in the tail of the signal, and all other shape information, is discarded. This is an interesting hypothesis to test, and if the model performs poorly compared to others which do take this additional information into account, this tells us that there is information in the signal other than the simple height of the peak value. Since this reduces each sensor trace to a scalar, the Peak Height model reduces the full dataset to 18 dimensions (one dimension per sensor).

The Lorentzian Model and unexpected sensor behaviour

Feature extraction was also performed using the Lorentzian model [Carmel et al., 2003], which is a model with 4 parameters. The Lorentzian model is specifically used for feature extraction of electronic nose data, so is less general than methods such as PCA but may be interpreted as using prior information about underlying physics of the measurement process. We thus have reason to expect good results.

The form for the Lorentzian model is as follows. $R_i(t)$ is the response of the ith sensor at time t.

$$R_i(t) = \begin{cases} 0, & t < t_i, \\ \beta_i \tau_i \tan^{-1}\left(\frac{t-t_i}{\tau_i}\right), & t_i \leq t \leq t_i + T, \\ \beta_i \tau_i \left[\tan^{-1}\left(\frac{t-t_i}{\tau_i}\right) - \tan^{-1}\left(\frac{t-t_i-T}{\tau_i}\right),\right], & t > t_i + T \end{cases} \tag{2.2}$$

This is illustrated in **Figure 2.28**. The 4 parameters of the Lorentzian model are t_i, β_i, τ_i and T. These are readily interpretable, which helps with setting initial

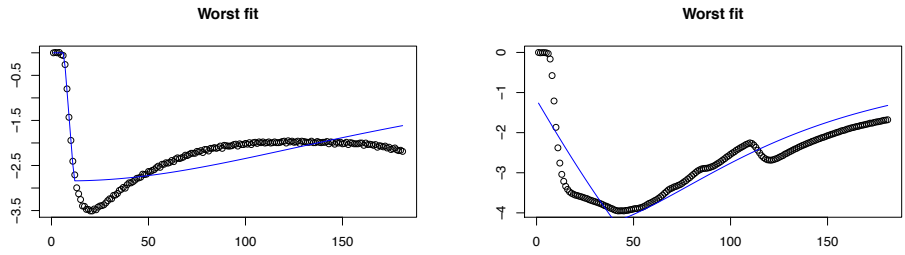


Figure 2.29: *Many of the sensor time-series do not behave as expected, and thus are fit poorly by the Lorentzian model. Here the worst fitting time-series are given. Left: Dry. Right: KOH.*

values:

1. t_i is the time the gas being analysed starts to pass over the sensor.
2. β_i is related to the amplitude of the signal. The maximum value attained in the Lorentzian model is $R_i(T) = \beta_i \tau_i \tan^{-1}\left(\frac{T}{\tau_i}\right)$.
3. τ_i is the decay time of the signal.
4. T is the time at which the signal reaches its maximum value, and is also the time at which gas stops being passed over the sensor.

The Lorentzian model was fitted to each 1d sensor time-series by minimising the sum of squares error using the R package `optimx` [Nash and Varadhan, 2011]. This is quite sensitive to the initial guesses of the 4 parameters of the model, but the interpretability of the parameters helped formation of a sensible initial guess.

The Lorentzian model did not fit the data well as it assumes that after the gas has stopped being passed over the sensor, the sensor value returns to 0. This is not the case since many of our sensors over-shoot their original value. Because of this we discard the tail of the data; The Lorentzian model is only fit up until the point where the sensor response has returned to 50% of its peak value. Some of our fits are poor (**Figure 2.29**), others are quite good if we do not consider the tail of the sensor trace (**Figure 2.30**).

Comparing the sensor traces for the best and the worst fitting Lorentz model cases, it can be seen that the possible variation between sensor traces may be quite complex without an obvious parametric form. This justifies the use of more flexible techniques such as ICA, which is considered below.

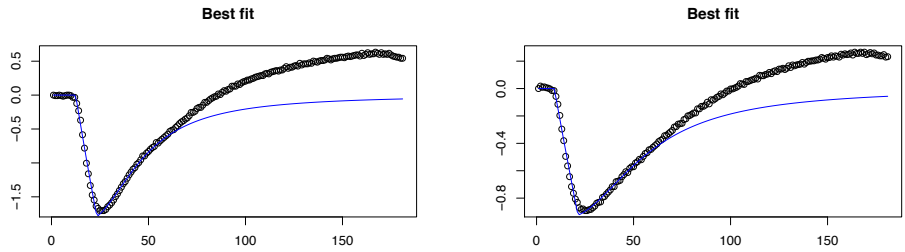


Figure 2.30: *Ignoring the tail (values after the trace has returned to 50% peak value), some sensors are well described by the Lorentzian model. The best fitting sensor time-series' are given for the dry (left) and KOH (right) data. The tail does not fit well because the Lorentzian model (blue line) cannot cross 0, so we have only fit the data up until the signal has decreased to 50% of its peak value.*

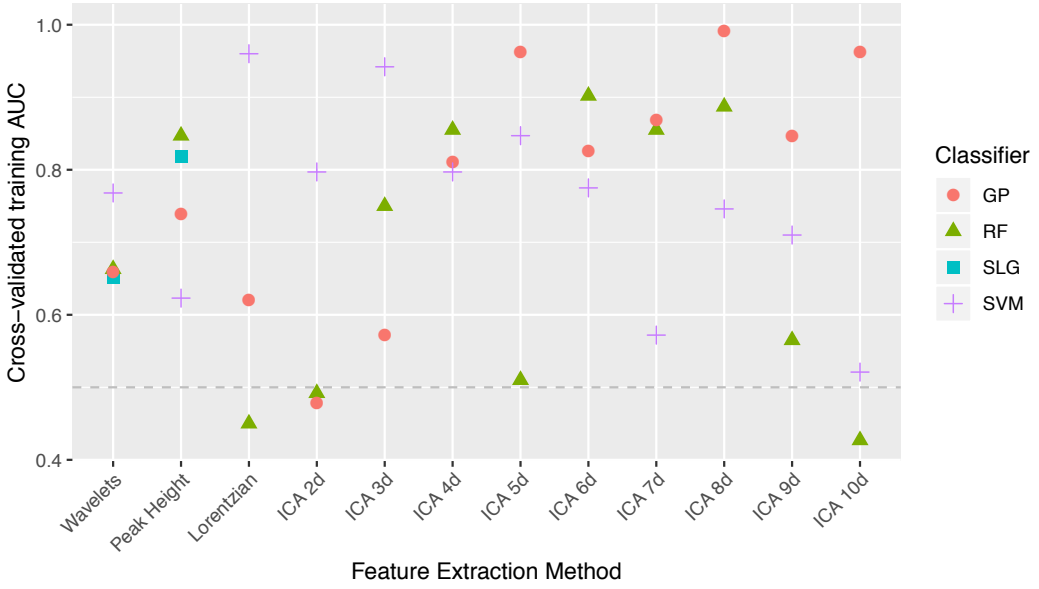


Figure 2.31: *AUCs achieved on training set using leave-one-out cross-validation for all pairs of feature extraction methods and classifiers under investigation. 4 dimensional ICA is selected using this plot; this gives the most stable performance and the greatest minimum AUC. The Random Forest classifier is also selected. Some points for the Sparse Logistic Regression classifier are absent because identical predictive probabilities were produced for all samples, and thus a meaningful ROC could not be computed.*

Training set model selection

Using the training dataset, we select a dimension reduction and classification pipeline. We attempt 4 different classification techniques: the Lorentzian model, wavelets paired with supervised feature selection, ICA and Peak Height. The dimension reduction is selected pairwise with one of the classifiers: Sparse Logistic Regression, Random Forest, Gaussian Process Classifier and Support Vector Machine. Selection is done by investigating AUCs produced in cross-validation on the training dataset for classifying candida vs non-candida. AUCs are compared in **Figure 2.31**.

The Lorentzian model and Peak Height are both described above. The wavelet transformation is a 1 dimensional version of that used in the FAIMS pipeline (**Section 2.2.2**), using the same Wilcoxon rank-sum supervised feature selection to 2 dimensions within the cross-validation stage.

The feature learning stage selected is ICA to 4 dimensions, which gives the most stable AUCs in **Figure 2.31**, and also the greatest minimum AUC. The Random Forest classifier is selected due to being the best performing classifier given the 4d ICA dimension reduction. This happens to be the same pair of methods as selected in the previous E-nose study done for this thesis (**Section 2.3.5**), though further investigation would be required to show if this is meaningful or not. We do not simply select the highest performing feature extractor/classifier pair since this is a small sample study and there is a high variance on these AUC estimates; we opt instead for a stable choice.

High training AUC variance

There is large variance in the LOO-CV training AUCs (**Figure 2.31**). Further investigation shows that the `fastICA` package used to perform ICA is nondeterministic, and predictive performance is sensitive to the seed used in fitting. This may explain some of the variance in AUC between different latent dimensionalities for ICA.

The high variance between classifiers within a single feature extraction method is more puzzling, but may be explained by the small training data set size (6 candida, 8 non-candida), and the fact that 3 of the classifiers used are highly flexible, so may exhibit high variance.

The high-variance AUC estimate makes it difficult for us to choose a feature exctraction method and classifier. Since the goal here is to select a model (as opposed to evaluating performance), using bootstrap resampling instead of LOO-CV may reduce the variance of the estimate [Kuhn and Johnson, 2013, Section 4.7]

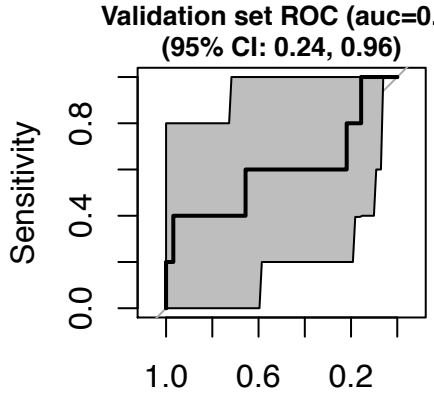


Figure 2.32: *ROC curve for candida/non-candida classification obtained by learning an ICA feature space transformation followed by a Random Forest classifier trained on the full training data, and applied to validation data. The mean AUC is not clinically significant, but there is also large uncertainty on the AUC estimate due to small samples sizes. Since the 95% confidence intervals on the AUC contain 0.5, we cannot confidently say that the FOX4000 with the statistical pipeline developed here is able to distinguish at all between candida positive and control swabs.*

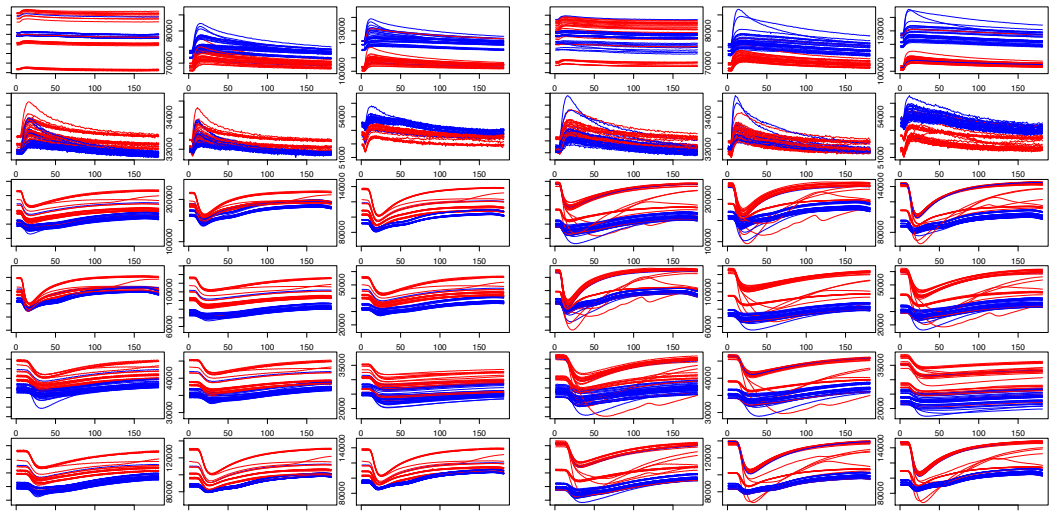
and allow us to to better pick a model, especially if the feature learning stage is performed each bootstrap.

Results on Validation Dataset

We use the validation dataset to evaluate the performance of the selected statistical pipeline. The training and validation datasets are pre-processed as above. It is important that the training and validation sets are scaled separately due to batch effects, which are discussed below. This is statistically valid to do since we are not using the disease label information in performing the separate standardisation.

We transformed the validation data to a 4 dimensional feature space by applying the same ICA transformation that we learned from (and applied to) the training data above. Then, using the Random Forest classifier trained on the feature representation of the full training dataset, we produce predictive probabilities for each feature representation of the validation set coming from a sample with candida. Combining these predictive probabilities with the known disease labels, we produce the ROC curve presented in **Figure 2.32** and obtain an AUC of 0.6 (95%CI 0.24–0.96).

Since 0.5 is well within the 95% confidence interval on the AUC, we cannot confidently say that the E-nose with our pipeline is able to distinguish between swabs from patients with and without candida.

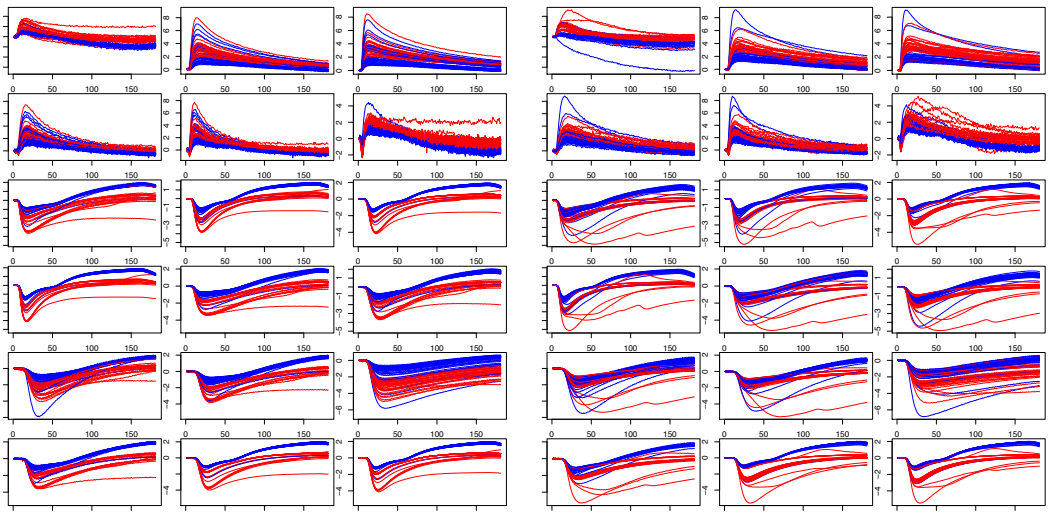


(a) *Sensor responses for dry samples* **(b)** *Sensor responses for KOH samples*

Figure 2.33: *There is a visual difference between the sensor responses on the training data (red) and validation data (blue). Each plot shows the responses of all sensors on all samples. Plot* ***Figure 2.33a*** *shows the sensor responses for the dry samples, and* ***Figure 2.33b*** *shows those for the KOH samples. The difference between training and validation data is most noticeable on the KOH samples.*

It appears that our statistical power is low, which we expect since the sample size is small. This produces large uncertainty on the AUC in **Figure 2.32**, and we make the decision not to pursue attempts at prediction any further. Although there are more questions of interest to ask this dataset, this kind of multiple hypothesis testing can be dangerous. By repeatedly coming up with new hypotheses (e.g., ability to discriminate bacterial from fungal infections) and testing if the hypothesis holds true, we may end up with a result that looks positive but is simply a statistical fluctuation. Thus, even if we were to get good results after more iterations on the statistical pipeline, confidence in the result would be low due to multiple hypothesis testing. Such a process is known as "p-hacking", where a researcher may, intentionally or not, keep generating hypotheses until a statistically significant p-value is found. The final result is then often published without detailing the process that has taken place, producing a published result that cannot be replicated. This is a known issue with a large amount of literature, and is discussed by Ioannidis [2005].

Investigating batch effects



(a) *Sensor responses for dry samples* **(b)** *Sensor responses for KOH samples*

Figure 2.34: *The separation between training and validation samples remains after standardisation. The change in sensor response between batches was more than just a shift and a scale. E.g., in row 4, column 1, it can be seen that the validation set curves (blue) end further from the baseline than the training curves.*

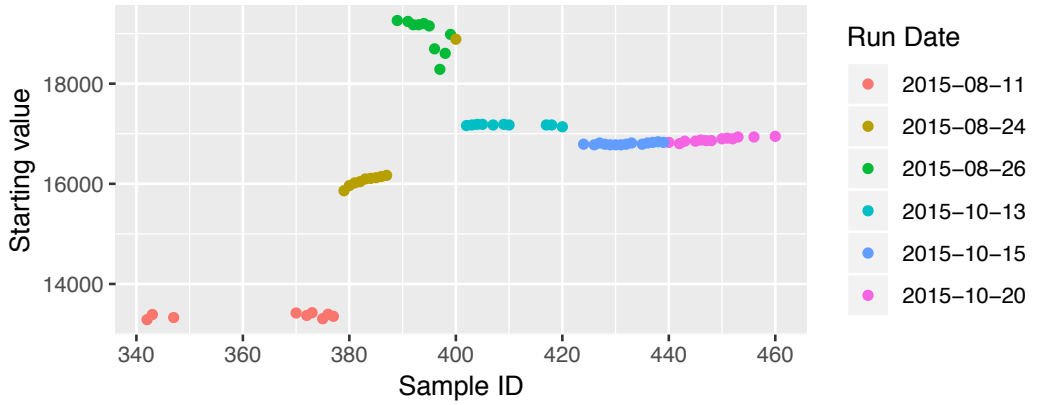


Figure 2.35: *Samples run in different batches have different baselines (sensor response values before a sample is measured). The y-axis shows the baseline value for sensor 1 (top left in **Figure 2.34**, **Figure 2.33**). The x-axis is the hospital assigned sample ID. Points are coloured by the day on which they were measured by E-nose. A clear nonlinear change in baseline between batches can be seen.*

Investigation reveals batch effects which make it challenging to generalise from the training dataset to the validation dataset. By plotting the raw time-series of every sensor for every sample in **Figure 2.33**, and colouring by membership of the training/validation dataset, a separation can be seen. It is not known what causes this separation between training and validation dataset, but it may be due to differences in experimental procedure, differences in background signal, or sensor contamination/drift.

The batch effects appear mostly as a shift of entire sensor responses by a fixed amount over time. Using the same standardisation procedure as above such that all sensors start at the same value removes this, producing **Figure 2.34**. The training data still reach a smaller peak on average than the validation data, despite being standardised separately. This is due to the unexpected overshoot of the sensors when returning to their initial values contributing to the standard deviation. Compensating for such batch effects thus requires a more sophisticated model. This illustrates another difficulty in practical implementation of electronic nose systems for diagnosis: the response to the same headspace may be different at different times.

Investigating sensor drift

The baseline value of the sensors at time 0 – before the gas to be analysed is presented – changes between samples. This is surprising and may affect our ability to generalise, since the effect is not just limited to shifting the sensor response but also changes the trajectory of the return to baseline value.

The batch effects are not limited to the training and validation sets, but affect the sensors on a daily basis. This is illustrated in **Figure 2.35**, which shows that samples run on the same day have similar baselines, and this baseline changes per day. Points are coloured by the day they were run, and the x-axis shows the IDs assigned to each sample by the hospital; it appears samples were likely run sequentially by ID. The y-axis shows the starting value of the sensors for dry samples. Clearly the sensor baseline changes day by day, and may also drift to a lesser extent throughout the day. It appears likely that the sample with ID 400 has been mislabelled as being run on 24/8/15.

There is more consistency in the validation set than the control set, with almost no change in baseline between the batch run on 15/10/2015 and that run on 20/10/2015. If there was a change in experimental process around this time it would be interesting to know what it was. It is not known what causes this change in baseline, but it is clearly nonlinear. It is also the case that the majority of the change occurs whilst experiments are not being run.

Conclusions

A study was performed to distinguish Candidiasis-positive patients from Candidiasis-negative patients using E-nose measurements of vaginal swabs. A prediction pipeline constructed on a batch of samples did not generalise well to a second batch of samples. Further investigation revealed large batch effects which would explain this poor generalisation.

Large uncertainty was repeatedly encountered. Model selection was difficult, with large variation in predictive accuracy depending on the classifier used. The validation AUC produced by the final model had wide 95% confidence intervals including AUC=0.5.

It was found that the sensors do not behave as expected. After the gas being analysed is no longer present, many sensors fail to return to their baseline value and may under or over-shoot the baseline. This means the Lorentzian models fail to fit the data. The baseline value also changes over time, affecting the sensor behaviour, meaning that correcting for this difference is not trivial. The change in baseline value is not a simple function of time or experiments run, and may involve experimental procedures or environmental conditions, so may be difficult to correct for.

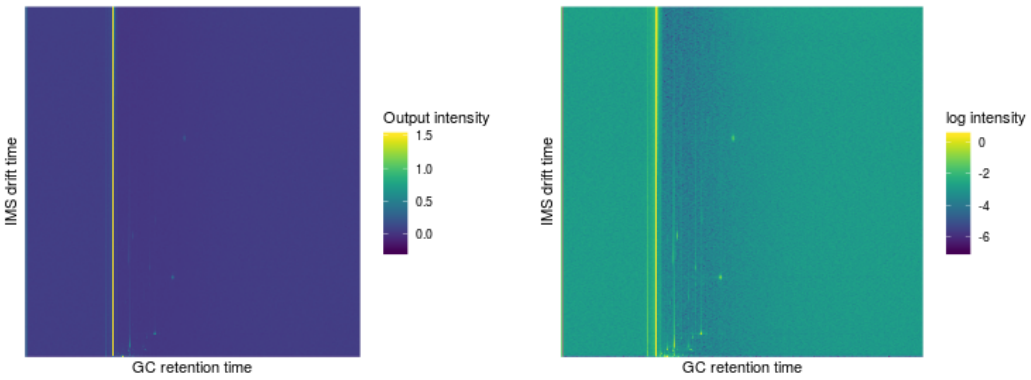


Figure 2.36: *Left: Raw output from the Flavourspec measurement of a single sample. Right: By taking the logarithm of the output more detail in the form of multiple peaks can be seen.*

2.3.7 Discriminating Diabetes from Obesity

This section describes the first analysis performed with the Flavourspec GC-IMS instrument (**Section 2.2.3**) at the University of Warwick. The experimental and data analysis parts of the analysis are thus largely exploratory, and a simple investigation of the data was performed before attempting more complex techniques.

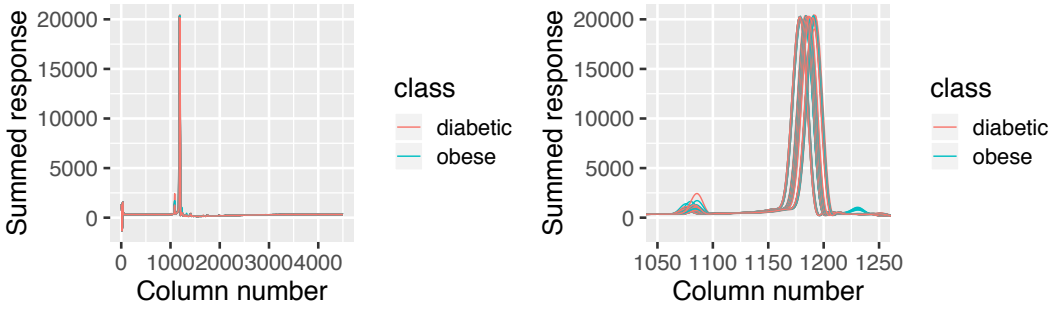
This was a small sample study of 56 urine samples (29 diabetic, 27 obese). The goal of the study was to discriminate between the diabetic and obese patients using urine headspace measurements. No training/validation data split was used since doing this previously on similarly sized data sets lead to problematically low statistical power. Instead, cross-validation was used to evaluate performance.

Due to the similarity of the structure of the Flavourspec data with that of the FAIMS data, a statistical pipeline similar to that developed for the FAIMS was produced. The high dimension of the Flavourspec data (25713000 features) meant that additional computational considerations were required.

The data

The Flavourspec measures a gas by first performing Gas Chromatography (GC) followed by Ion Mobility Spectrometry (IMS). The molecules in the gas are first separated by GC. This 1d separation of the original gas is then discretised into 5714 compartments, and each compartment is measured by IMS at 4500 points. This produces a 2d grid of positive real valued IMS signal intensity. The output of the Flavourspec for single sample is illustrated in **Figure 2.36**.

Each sample is $5714 \times 4500 = 25713000$ dimensional; representing each value as a double precision (8 byte) floating point number means the data are $25713000 \times$



(a) **(b)**

Figure 2.37: ***(a):*** *Summing up each of the columns of a single Flavourspec measurement allows a single sample to be represented by a single curve, allowing the visual comparison of many samples.* ***(b):*** *Zooming the x-axis on the region of largest signal variation, a "bump" around column number 1230 can clearly be seen which may potentially distinguish the diabetic and obese samples.*

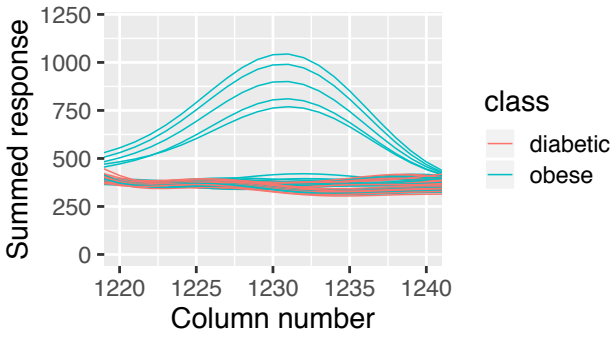


Figure 2.38: *Zooming further into the "bump", it can be seen that only 5 of the 27 obese samples are in the bump.*

8bytes $\approx$ 0.2GB per sample. The full dataset of 56 samples is thus 11.5GB, which is large enough for the dataset to require special treatment when loading into memory to avoid computational difficulties on a modern laptop.

Investigating the dataset

The dataset is first investigated for any simple and clear structure before moving on to more advanced techniques. Learning about the distribution of diabetic and obese samples by visually comparing the 2d images (such as **Figure 2.36**) of all 56 samples is difficult. By summing up along the columns a 1d curve is obtained, which is simpler to compare between samples as multiple curves may be plotted. This is shown in **Figure 2.37a** where curves are coloured depending on if the corresponding sample was from a diabetic or obese patient. **Figure 2.37b** shows the same plot with the x-axis restricted to the region in which most of the signal variation occurs.

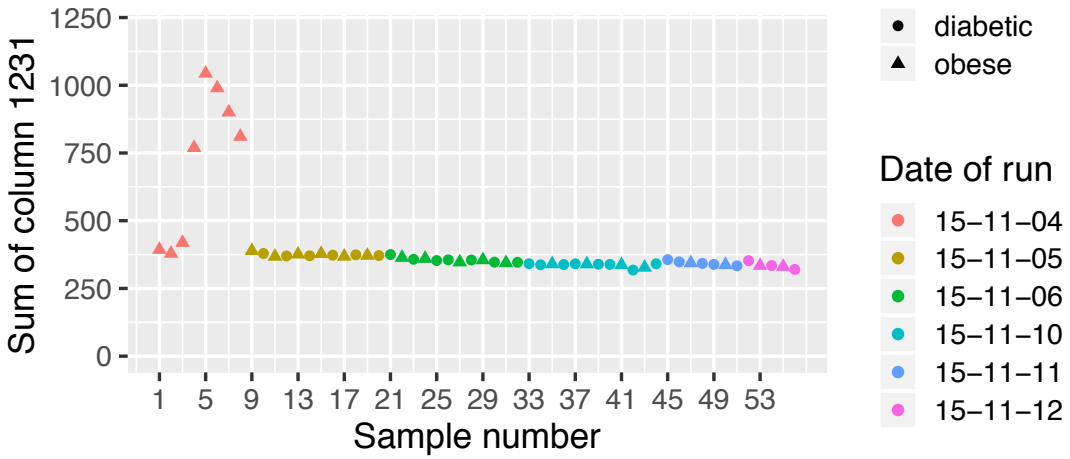


Figure 2.39: *Looking at the bump height of each sample, it can be seen that the bump was likely an experimental artefact which disappeared overnight. A slow linear drift towards reduced sensor response is also apparent.*

To the right of **Figure 2.37b** a bump of blue (obese) curves can be seen. Zooming in on this region we obtain **Figure 2.38**, revealing that the bump does not distinguish diabetic from obese samples; only 5 of the 27 obese samples are in the bump. By considering only column 1231 (containing the top of the bump), the bump height of each curve can be plotted on the y-axis, and samples ordered along the x-axis by the order in which they were run. This gives **Figure 2.39**, where points have been coloured by the day the sample was run, and the point shape shows the class of the sample. It can be seen that all samples forming the bump (sample numbers 4–7) were run sequentially and on the same day. The bump is likely due to a factor affecting the experiment; this could be sensor contamination, a change in background signal or any other unknown factor. Importantly it seems unlikely that this bump is predictive of anything we might be interested in.

Figure 2.39 also shows that the sample classes are well mixed after the first day when only obese samples were run. There also appears to be a slow, linear drift of reduced sensor response over the 7 days the experiment was being run. This drift appears to depend more on the number of samples run than the time that has passed, since there is no jump between the samples run on 06/11/15 and those run on 10/11/15. If the sample classes were mixed on the first day, a diabetic sample would likely have fallen within the bump. Depending on whether this hypothetical point also formed part of the bump, this would have provided strong evidence either for or against the hypothesis that the bump is non-predictive.

The change in the mean signal over time was investigated by summarising each sample as the mean of the sensor response, giving **Figure 2.40**. This reveals

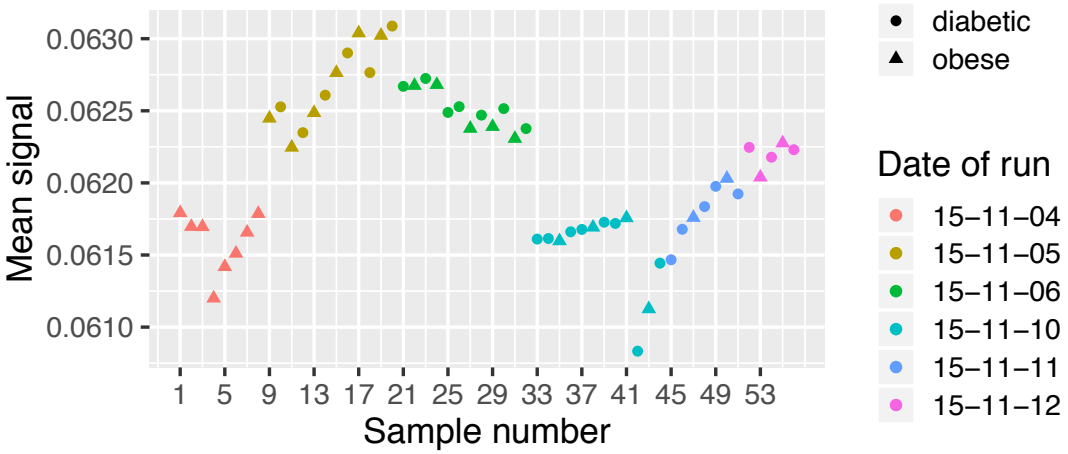


Figure 2.40: *The mean signal from the Flavourspec output for each sample is plotted. Samples are placed on the x-axis in the order they were run. The mean signal drifts over time, with large jumps between days. There is no obvious difference in the mean signal between diabetic and obese samples.*

clear batch effects as well as a within-batch drift. Such batch effects can make predictions difficult. However, given flexible machine learning techniques and data from sufficiently many batches, it can still be possible to learn a model able to produce accurate predictions in spite of batch effects.

Adapting the wavelet decomposition pipeline

A statistical pipeline has previously been developed for the FAIMS based on performing a wavelet decomposition to the smooth 2d FAIMS data and using the wavelet coefficients as features. Inside a cross-validation loop, 2 of the wavelet coefficients are selected using the Wilcoxon rank-sum supervised feature selection technique, and these features are used to train a classifier and predict the class of the out-of-fold points.

The FAIMS data and the Flavourspec data have common structure in the 2d, smooth representation of each sample. Thus, there is reason to believe that the FAIMS pipeline, when adapted to the Flavourspec data, may perform well.

Loading and decomposing the data

For computational reasons we chose to sequentially load each sample, performed a wavelet decomposition on that sample, and set any wavelet coefficients smaller than a threshold to zero, giving a sparse feature representation of the sample. To pick the threshold we loaded a single sample and decomposed it into wavelet coefficients.

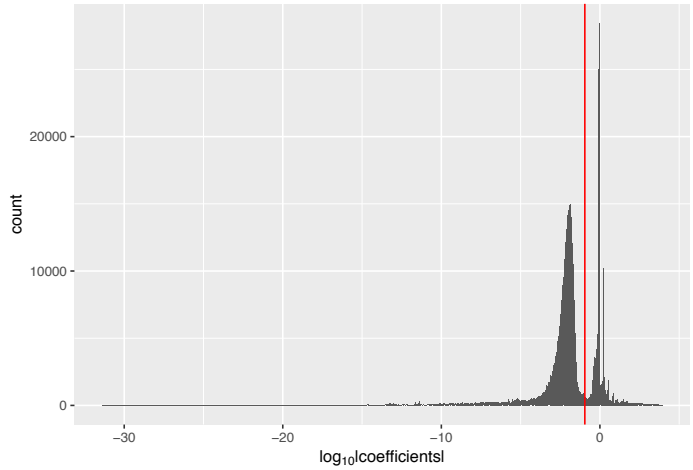


Figure 2.41: *A Histogram of the log absolute values of the wavelet coefficients used to represent a single sample shows bimodality. A threshold of* $10^{-0.95}$ *is chosen (red line) and all wavelet coefficients with a magnitude smaller than this are set to zero.*

Showing a histogram of the log of these coefficients in **Figure 2.41**, a bimodality can be seen where wavelets contributing a lot to the Flavourspec measurement are separated from those which contribute very little. A threshold is selected at the trough between the two modes.

The number of wavelet coefficients before thresholding is 1398101. We set all wavelet coefficients with absolute values less than $10^{-0.95} \approx 0.11$ to zero, producing a sparse matrix with 160334 non-zeroes. This is 11.47% of the original dimensionality, requiring 1.2Mb per sample.

The appropriateness of this level of compression was checked by reconstructing the 2d Flavourspec output from the sparse coefficients, which is illustrated in **Figure 2.42**. These are visually very similar to the raw data in **Figure 2.36**, and by eye there are no important features that have been lost. Comparing the raw data to that recovered from the sparse wavelet coefficients, the root mean squared pixel error is 5.5×10^{-3}, which is small compared to the mean pixel intensity of 0.83.

All samples were then loaded into memory, sequentially being wavelet decomposed, thresholded, and added to a sparse matrix containing all samples. Any columns of this matrix (corresponding to coefficients for a single wavelet) containing fewer than 6 non-zeroes were removed. This removed 2833 columns, leaving 11125 dimensional data which can comfortably fit in memory on a modern laptop. We also dropped any columns with a standard deviation below $10^{-1.9}$, dropping a further 6101 columns. This is a sensible feature selection step as we do not expect regions of the Flavourspec measurement plane which vary very little to be informative with respect to sample classification. A histogram of column standard deviations along

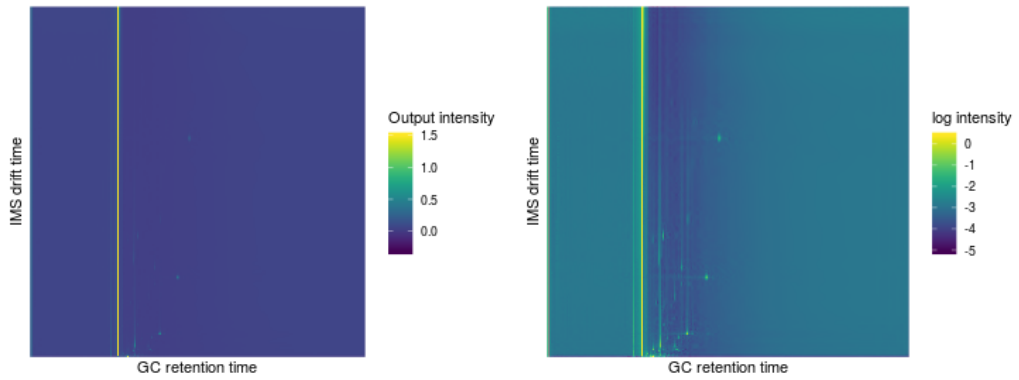


Figure 2.42: *To check the suitability of the threshold chosen, the measurement has been recreated from the thresholded wavelet coefficients. Comparing this to* ***Figure 2.36****, no important features have been lost.*

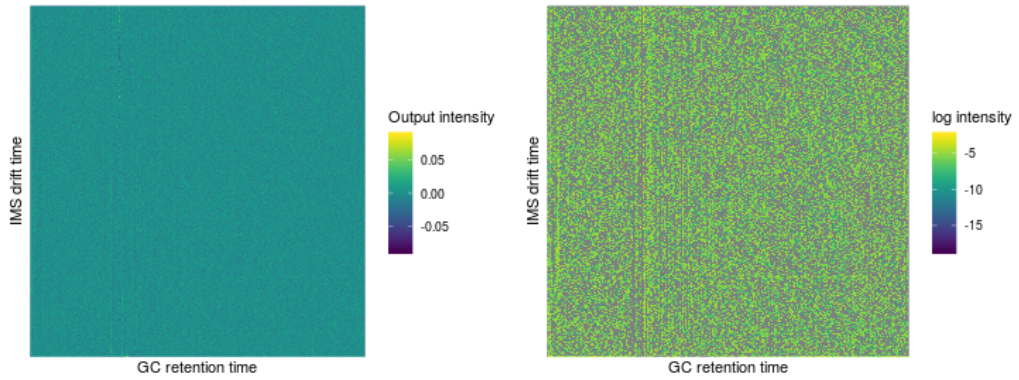


Figure 2.43: *Here we plot the residuals between the wavelet reconstructed data in* ***Figure 2.42*** *and the original data in* ***Figure 2.36****. No structure of importance can be seen in the residuals.*

with the threshold is given in **Figure 2.44**.

Predictive accuracy

After being processed as above the data were of dimension 149683. Due to the small sample size of 56, and having been affected by low statistical power in previous similarly sized studies, we opted not to use a separate validation set and instead 10-fold cross-validated on the full dataset to produce predictive probabilities.

As with the pipeline developed for the FAIMS (**Section 2.2.2**), this cross-validation includes a supervised feature selection stage. The feature selection stage happens before the classification stage, and uses only the labels from the training data. For each feature, the R implementation of the Wilcoxon rank-sum test [R Core Team, 2014] is used to test the null hypothesis that the diabetic and obese samples

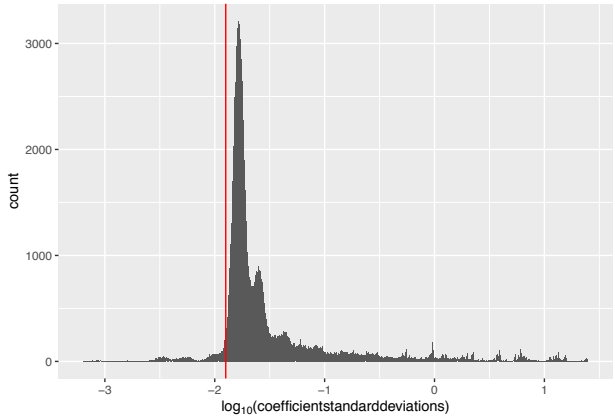


Figure 2.44: *Histogram of standard deviations logarithms (base 10) of wavelet coefficients. This is constructed after removing any wavelets with fewer than 6 samples with absolute coefficients above the threshold of* $10^{-0.95}$*. The red line shows another threshold; any wavelets with standard deviations below* $10^{-1.9}$ *are dropped.*

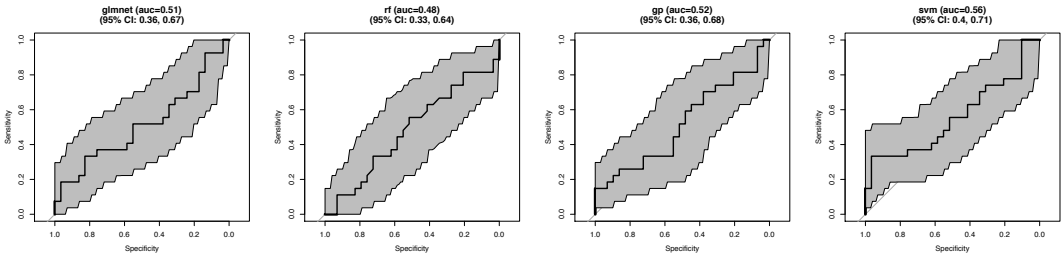


Figure 2.45: *AUCs obtained in a 10-fold cross-validation loop of statistical pipeline developed for Flavourspec. These particular ROCs retain 5 features prior to classification, however varying this from 2 to 12 does not significantly change the ROC curves. No statistically significant ability to predict the sample class was discovered.*

were selected from populations having the same distribution [Hollander et al., 2013]. This produced a p-value for each feature, and we dropped all features except those with the smallest p-values. That is, we retained only the features which appear to be the most discriminative. The number of features to retain was varied from 2 to 12.

Finally, to produce predictions we used the set of 4 classifiers: Random Forest [Liaw and Wiener, 2002], SVM [Karatzoglou et al., 2004], Sparse Logistic Regression [Friedman et al., 2010] and Gaussian Process Classifier [Rasmussen and Williams, 2005; Karatzoglou et al., 2004]. We were not able to achieve any statistically significant or clinically significant AUCs from any pair of feature learning techniques and classifiers; all were similar to those given in **Figure 2.45**.

Conclusions

During this study no model was found that could accurately discriminate between samples from diabetic and obese patients in cross-validation. If there is any signal in the data, we have not been able to discover it. It may be the case that this was due to a flaw in the experimental procedure.

Using 2d wavelet transformation and coefficient thresholding we have been able to compress the Flavourspec data whilst minimally impacting the measurement reconstructions. However, we do not know if this will generalise to samples containing greater signal.

2.3.8 Summary

This section has detailed the analyses on artificial olfaction data performed for this thesis. The subsections are presented in chronological order of the analyses. The Whiskey study was performed first, which is the only study in which the entire process—pipetting samples and operating the FAIMS—was performed by the author. This was an illuminating exercise, highlighting a number of unexpected sources of variance: consistency of pipetting and dilution quantities, sensor contamination from previous samples and variance in decontamination times (the first sample in the day is effectively uncontaminated), and ambient temperature/humidity/odours. It was interesting that the whiskey that the statistical pipeline was best able to discriminate from the others was that which most humans would also also find the most unique (the peated whiskey). The fact that 100% classification accuracy was achieved in that case indicates the potential strong discriminative potential of FAIMS.

FAIMS was investigated as a possible method of diagnosing IBD, and for distinguishing between two IBD sub-types. Breath from 53 IBD patients (29 UC, 24 CD) and 11 healthy volunteers was collected in an IBD clinic and transported to Warwick University for analysis. For discriminating between the IBD-positive patients and healthy controls we obtained an AUC of 0.82. For distinguishing between IBD sub-types we obtained an AUC of 0.68. The moderate ability to distinguish subtypes indicates that disease-specific signal exists within the FAIMS measurement, and the ability to diagnose IBD is not based entirely off some general indicator of poor health, such as inflammation or immune response.

Breath samples of 21 TB patients and 19 healthy controls were collected, and an AUC of 0.83 produced in classifying the samples within leave-one-out cross-validation.

A study into Hepatic Encephalopathy diagnosis was performed, with 13 covert HE, 9 overt HE, 20 healthy control samples. Breath samples were collected within the hospital and analysed with uvFAIMS. The standard pipeline lead to ability to distinguish HE samples from control samples with an AUC of 0.84, and discrimination between the HE sub-types with an AUC of 0.71. Including additional unlabelled FAIMS data in the feature learning stage increased the AUC to 0.96. This was a relatively small sample study, so error bars on all the AUCs are quite large.

A larger study into type 2 diabetes was performed using the E-nose to analyse the headspace of patient urine. Urine samples were collected from 41 type-2 diabetic patients and 48 healthy control volunteers. This study initially looked promising, producing an AUC of 1 on a held-out validation set. However, it was later discovered that the control samples and the diabetic samples had been frozen for different periods of time, and the pipeline had likely only learned to classify samples based off the amount of VOC degradation having occurred in storage, and not based off any disease-specific signal. Useful lessons were still learnt from the study: useful signal is likely spread across multiple sensors, it is important to consider possible confounders in study design, and a validation set should ideally come from an independent experiment.

A set of vaginal swabs were taken from patients with Bacterial Vaginosis, Group B Streptococcal infection and Candidiasis. These were collected and analysed with the E-nose in two batches, which were used as a training dataset and a held-out validation set. A predictive pipeline was trained on the training set but did not generalise to the validation set, which was determined to be due to large batch effects. Further investigation may have been interesting, but small sample size and low statistical power meant searching for results would likely have uncovered false-positive conclusions.

Urine from diabetic and obese patients was taken to be analysed by the Flavourspec—a GC-IMS instrument. The Flavourspec output is extremely high dimensional and requires special considerations because of this. Unfortunately, the data do not appear to contain any appreciable signal for distinguishing between the two classes. However, a pipeline similar to that for the FAIMS (**Section 2.2.2**) was developed, which appears to compress the data with little signal loss, and may be used in later studies.

Chapter 3

Structured PCA: Theory

This chapter introduces the novel unsupervised learning method which we call Structured PCA (StPCA). This model is closely related to PCA, and builds upon the Probabilistic PCA (PPCA) model introduced by Tipping and Bishop [1999]. An implementation of StPCA has been made available in the R package `stpca` hosted at `https://github.com/JimSkinner/stpca`. Documentation for this package has been included in **Appendix C.2**.

Given a set of d dimensional data, StPCA learns a latent representation of the data, meaning each sample is represented in $k < d$ dimensions. In doing this, StPCA also learns a covariance structure across the measured variables. The $d \times k$ *loadings matrix* $\mathbf{W}$ will be the focus of much of the chapter, and plays the role of mapping the k-dimensional latent space to the d-dimensional observation space. As we will see, $\mathbf{W}$ also describes the covariance structure modelled by StPCA, and the columns of $\mathbf{W}$ are closely related to the Principal Components (PCs) in PCA.

The novel contribution of StPCA is that the user may specify a Gaussian Process prior over the loadings, which reduces to specifying a covariance function. It is assumed that each dimension is associated with a 'location' variable, such as the relative location of pixels in an image. A Gaussian posterior is then inferred and Bayes factors may be computed, enabling model selection for the value of k and the choice of covariance function used, as well as tuning of covariance function hyper-parameters.

StPCA can play a role in an analysis similar to that of PCA. We use StPCA in this thesis as a feature extraction method. High dimensional samples are reduced in dimension by considering their latent representations as learned by StPCA. As well as moving to a latent space, the first few columns of the loadings – analogous to the PCs – can also be of interest to a researcher since they inform about the

structure of variations in a given problem.

StPCA also enables a Normal approximation to the posterior over the loadings. By examining the covariance of this uncertainty reported by StPCA, a researcher can better quantify the certainty of any conclusions drawn and understand to what extent the covariance structure is well specified by the data. In contrast, classical PCA produces only a point estimate of the PCs. StPCA would thus be helpful in cases where we would like to quantify our uncertainty.

StPCA is a probabilistic, graphical, linear latent variable model paired with an efficient algorithm for approximate posterior inference. This algorithm is deterministic and iterative, and does not use rejection sampling or other Monte Carlo techniques for inference. Instead, approximations are made, producing a Gaussian approximation to the posterior, and tractable computations.

This chapter is structured as follows. **Section 3.1** formally introduces the StPCA model and discusses how the Laplace approximation may be used to compute an approximate posterior and evidence, enabling uncertainty quantification and model selection. **Section 3.2** goes in to more detail about how to construct the prior, and its relation to Gaussian Processes and covariance functions. **Section 3.3** derives the computations required to perform inference. **Section 3.4** discusses how PCA may be recovered as a special case of StPCA, and **Section 3.5** concludes.

3.1 Model

A design matrix of zero-mean observations $\mathbf{X} = [\mathbf{x}_1 \cdots \mathbf{x}_n]^\top \in \mathbb{R}^{n \times d}$ is assumed provided. The number of observations is denoted n, and the dimensionality of the observations is d. Each $\mathbf{x}_i$ for $i \in 1 \cdots n$ is a column vector of length d representing a single sample.

In the StPCA model, each $\mathbf{x}_i$ is paired with an unobserved latent variable $\mathbf{v}_i \in \mathbb{R}^k$ where $k < d$ is the dimensionality of the latent space. $\mathbf{V}$ is the $n \times d$ matrix where the ith row is $\mathbf{v}_i$. The mapping from latent space to observation space is linear and represented by the parameter $\mathbf{W} \in \mathbb{R}^{d \times k}$, the ith column of which is referred to by $\mathbf{w}_i$. The parameter σ^2 controls the magnitude of the noise in the modelled data generating process. Each feature of $\mathbf{X}$ is associated with a location variable, where the location for feature i is denoted $\mathbf{t}_i$. Prior covariance between features is specified with a covariance function acting between each pair of features, producing the $d \times d$ covariance matrix $\mathbf{K}_\beta$. Covariance function hyper-parameters are denoted β.

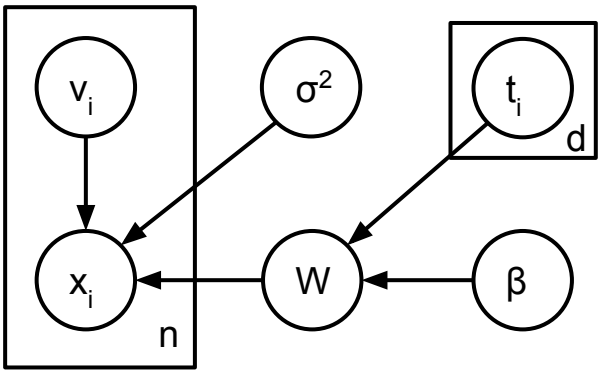


Figure 3.1: *Plate diagram for StPCA. This is equal to that for PPCA but with the addition of the hyper-parameters β and the location variables $\mathbf{t}_1 \dots \mathbf{t}_d$ on which only the loadings $\mathbf{W}$ depend.*

3.1.1 Exact model

The joint probability over all random variables in the exact model has the factorisation

$$p(\mathbf{X}, \mathbf{V}, \mathbf{W}, \sigma^2, \beta) = p(\mathbf{X}|\mathbf{V}, \mathbf{W}, \sigma^2)p(\mathbf{V})p(\mathbf{W}|\beta)p(\sigma^2)p(\beta) \tag{3.1}$$

$$= \big[\prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{v}_i, \sigma^2, \mathbf{W})p(\mathbf{v}_i)\big]\big[\prod_{j=1}^{k} p(\mathbf{w}_j|\beta)\big]p(\sigma^2)p(\beta) \tag{3.2}$$

which corresponds to the plate diagram in **Figure 3.1**. Although $p(\mathbf{W}|\beta)$ depends also on the location variables $\mathbf{t}_1 \dots \mathbf{t}_d$, this is omitted from the syntax – as are d and k – since these are fixed, non-random quantities.

Equation 3.2 contains the sub-product

$$\prod_{i=1}^{n} p(\mathbf{x}_i|\mathbf{v}_i, \sigma^2, \mathbf{W})p(\mathbf{v}_i) = \prod_{i=1}^{n} p(\mathbf{x}_i, \mathbf{v}_i|\sigma^2, \mathbf{W}) \tag{3.3}$$

$$= p(\mathbf{X}, \mathbf{V}|\mathbf{W}, \sigma^2) \tag{3.4}$$

which we call the *complete data likelihood*, as this is the joint likelihood of the data and the latent variables given all parameters. The distribution of a single $\mathbf{x}_i$ depends only on the paired $\mathbf{v}_i$, and does not change when all other $\mathbf{v}_{j\neq i}$ change. $\mathbf{v}_i$ is thus an unobserved latent representation of $\mathbf{x}_i$ with dimension $k \leq d$.

The model is defined fully by defining each of the terms in the factorisation.

$$p(\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{x}_i|\mathbf{W}\mathbf{v}_i, \sigma^2 I) \qquad i \in 1, \ldots, n \tag{3.5}$$

$$p(\mathbf{v}_i) = \mathcal{N}(\mathbf{v}_i|\mathbf{0}, I) \qquad i \in 1, \ldots, n \tag{3.6}$$

$$p(\mathbf{w}_i|\beta) = \mathcal{N}(\mathbf{w}_i|\mathbf{0}, \mathbf{K}_\beta) \qquad i \in 1, \ldots, k \tag{3.7}$$

$$p(\beta) \propto 1 \tag{3.8}$$

$$p(\sigma^2) \propto \sigma^{-2} \tag{3.9}$$

From **Equation 3.5**, the role of parameters $\mathbf{W}$ and σ^2 can be seen. If we know the model parameters and the latent representation of a sample, we obtain a distribution over what the observation could possibly have been. The mean of this distribution is obtained by applying the loadings $\mathbf{W}$ to project $\mathbf{v}_i$ up to dimension d. $\mathbf{W}$ is thus in $\mathbb{R}^{d\times k}$. There is σ^2 isotropic variance around this mean, which is the same for every sample.

The term $\mathbf{K}_\beta$ in **Equation 3.7** is the *prior covariance matrix* and is a function of the hyper-parameters β. We assume that each of the d dimensions of the observed data space is associated with a location parameter $\mathbf{t}_i, i \in 1 \cdots d$. $\mathbf{K}_\beta$ is then constructed by applying a covariance function $C(\cdot, \cdot\,; \beta)$ parametrised by β to obtain the prior covariance between every pair of dimensions. Symbolically, the i, jth element of $\mathbf{K}_\beta$ is constructed as:

$$(\mathbf{K}_\beta)_{ij} = C(\mathbf{t}_i, \mathbf{t}_j; \beta) \quad \forall\, i, j \in 1 \cdots d \tag{3.10}$$

As a simple example, say we are working with images such that each element of an observation is the grey-scale intensity of a pixel, and $\mathbf{t}_i$ is the location of pixel i. Choosing a covariance function such as the Squared Exponential then encodes the prior knowledge that nearby pixels have strong prior covariance, so are expected to take similar values. When performing inference, this translates into encouraging elements of each $\mathbf{w}_i$ associated with nearby pixels to take similar values. A single learned $\mathbf{w}_i$ will then capture a smooth image, and a sample reconstruction $\mathbf{W}\mathbf{v} = \sum_{i=1}^{k} \mathbf{w}_i^\top v_i$ is built from a linear combination of these underlying images.

The marginal priors $p(\beta)$ and $p(\sigma^2)$ are improper. We assume the domain of β is over the reals, so an uninformative uniform prior is appropriate. For σ^2, we use the Jeffrey's prior for a scale parameter [e.g, Jaynes, 2003, Chapter 12]. The uniform prior over β does not present any problems since we do not integrate over β at any point (we approximate the evidence with the marginal likelihood). We do integrate over σ^2 in computing the marginal likelihood, which is improper; however,

Variable	Domain	Meaning
n	$\mathbb{Z}^+$	Number of observations.
d	$\mathbb{Z}^+$	Dimensionality of observations.
k	$\mathbb{Z}^+, k \leq d$	Dimensionality of latent variables.
$\mathbf{x}_i$	$\mathbb{R}^d$	A single observation indexed by $i \in 1 \cdots n$.
$\mathbf{X}$	$\mathbb{R}^{n \times d}$	Design matrix of observations; the ith row is $\mathbf{x}_i^\top$.
$\mathbf{v}_i$	$\mathbb{R}^k$	Latent representation of $\mathbf{x}_i$, $i \in 1 \cdots n$.
$\mathbf{V}$	$\mathbb{R}^{n \times k}$	Matrix of latent variables. Row i is $\mathbf{v}_i^\top$.
$\mathbf{W}$	$\mathbb{R}^{d \times k}$	The *loadings matrix* which maps the k-dimensional latent space up to the space of d-dimensional observations.
σ^2	$\mathbb{R}^+$	The variance of the predicted distribution of an observation given its latent representation.
$\mathbf{K}_\beta$	$\mathbb{R}^{d \times d}$	*Prior covariance matrix* which encodes our prior knowledge of the covariance structure of our observations. Depends on hyper-parameters β.
β	$\mathbb{R}^{n_\beta}$	Hyper-parameters controlling $\mathbf{K}_\beta$.
$\mathbf{t}_i$	$\mathbb{R}^{n_\mathbf{t}}$	Location parameter associated with each of the d dimensions of the observation space, $i \in 1 \cdots d$.
$C(\cdot, \cdot\,; \beta)$	$\mathbb{R}^{n_\mathbf{t}} \times \mathbb{R}^{n_\mathbf{t}} \mapsto \mathbb{R}$	Covariance function, parametrised by β, used to construct $\mathbf{K}_\beta$: $(\mathbf{K}_\beta)_{ij} = C(\mathbf{t}_i, \mathbf{t}_j; \beta)\ \forall i, j \in 1 \cdots d$.

Table 3.1: *Variables used in StPCA*

we do not deal with the true marginal likelihood and consider only the Laplace approximation, the nature of which always produces a normalised distribution.

StPCA is related to the GP-LVM (**Section 1.1.1**), which also introduces a Gaussian prior on $\mathbf{W}$. The GP-LVM uses the prior $p(\mathbf{W}) = \prod_{i=1}^{k} \mathcal{N}(\mathbf{w}_i | \mathbf{0}, \alpha I)$, which can be also be constructed in StPCA using the independent covariance function. Using this prior and integrating out the loadings leads to a linear Gaussian Process mapping from latent to observaion space [Lawrence, 2004]. The difference in model between StPCA and GP-LVM is that the GP-LVM may extend to a non-linear latent-to-observation mapping by introducing a non-linear kernel in the Gaussian Process mapping. However, with a linear GP-LVM the models are quite similar. The inference procedures still differ in that StPCA marginalises out the latent variables and optimises the loadings, whilst the GP-LVM marginalises out the loadings and optimises the latent variables.

Calculating the likelihood

The likelihood of StPCA can be computed in closed form. Using **Equations 3.5** and **3.6**, we marginalise out the latent variables

$$p(\mathbf{x}_i|\mathbf{W},\sigma^2) = \int_{\mathbf{R}^k} p(\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2)p(\mathbf{v}_i)\ \mathrm{d}\mathbf{v}_i \tag{3.11}$$

$$= \int_{\mathbf{R}^k} \mathcal{N}(\mathbf{x}_i|\mathbf{W}\mathbf{v}_i,\sigma^2 I)\mathcal{N}(\mathbf{v}_i|\mathbf{0},I)\ \mathrm{d}\mathbf{v}_i \tag{3.12}$$

The same marginalisation is performed in PPCA [Tipping and Bishop, 1999, Equation 3]. **Equation 3.12** has the form of a marginalisation over the mean of a Gaussian with a conjugate Gaussian prior [Gelman et al., 2013, Section 2.4]. The marginal distribution is thus Gaussian, so we only need to compute the mean and covariance:

$$\mathbb{E}\big[\mathbf{x}_i|\mathbf{W},\sigma^2\big] = \mathbf{0} \tag{3.13}$$

$$\mathrm{cov}(\mathbf{x}_i|\mathbf{W},\sigma^2) = \mathbf{W}\mathbf{W}^\top + \sigma^2 I \tag{3.14}$$

The derivation of these is bulky and has been moved to **Appendix D.1.1**. Using these, this gives us the closed form likelihood

$$p(\mathbf{x}_i|\mathbf{W},\sigma^2) = \mathcal{N}(\mathbf{x}_i|\mathbf{0},\mathbf{W}\mathbf{W}^\top + \sigma^2 I) \tag{3.15}$$

Considering the covariance of this Gaussian, we see that k directions of variance are modelled by the rank-k matrix $\mathbf{W}\mathbf{W}^\top$, and each of these directions has variance of at-least σ^2 due to the addition of the $\sigma^2 I$ term. The remaining $d-k$ directions all have variance of exactly σ^2. If σ^2 is very small, the data are modelled as laying on a degenerate Gaussian spanning only a k-dimensional subspace.

Latent Space

We have seen from **Equation 3.5** that if we know the latent representation of a sample, one can obtain a distribution over possible observations (repeated here, for convenience):

$$p(\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2) = \mathcal{N}(\mathbf{x}_i|\mathbf{W}\mathbf{v}_i,\sigma^2 I)$$

However, if we have some observations and are interested in their unobserved latent representations, it is more natural to consider the posterior over latent variables:

$$p(\mathbf{v}_i|\mathbf{x}_i,\mathbf{W},\sigma^2) = \mathcal{N}(\mathbf{v}_i|\mathbf{M}^{-1}\mathbf{W}^\top\mathbf{x}_i,\sigma^2\mathbf{M}^{-1}) \tag{3.16}$$

which uses the definition $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2 I$. Note that the difference from $\mathbf{C} = \mathbf{W}\mathbf{W}^\top + \sigma^2 I$, being of order $k \leq d$. This may be computed by noting both $p(\mathbf{x}_i|\mathbf{v}_i)$ and $p(\mathbf{v}_i)$ are Gaussian and using the rules of Gaussian conditioning [e.g., Bishop, 2006, Section 2.3.3].

Non-identifiability of W and V

StPCA has a non-identifiability in $\mathbf{W}$ and $\mathbf{V}$. For any orthogonal matrix $\mathbf{R} \in \mathbb{R}^{k\times k}$, making the substitution $\mathbf{W} \to \mathbf{W}\mathbf{R}$ does not change the likelihood nor the prior (and thus the posterior). This is the case because $\mathbf{R}$, by definition, satisfies $\mathbf{R}\mathbf{R}^\top = \mathbf{R}^\top\mathbf{R} = I$, and $\mathbf{W}$ only appears in the likelihood and prior as $\mathbf{W}\mathbf{W}^\top$, which would be substituted as $\mathbf{W}\mathbf{W}^\top \to (\mathbf{W}\mathbf{R})(\mathbf{R}^\top\mathbf{W}^\top) = \mathbf{W}\mathbf{W}^\top$. The interpretation of this is that $\mathbf{W}$ defines a subspace, and one can always change basis within this subspace without changing the model.

Furthermore, when considering the complete likelihood in which $\mathbf{v}_i$ also occurs, these also appear only as the product $\mathbf{W}\mathbf{v}_i$, which also is invariant to the substitution $\mathbf{W}\mathbf{v}_i \to (\mathbf{W}\mathbf{R})(\mathbf{R}^\top\mathbf{v}_i) = \mathbf{W}\mathbf{v}_i$. $\mathbf{R}$ captures rotations around the origin in the latent space, as well as sign flips (reflections around the origin) and permutations of the columns of $\mathbf{V}$. This tells us that there is no preferred orientation of the latent space; one may always rotate and reflect the points in the latent space around the origin, as long as the projection $\mathbf{W}$ from the latent to observed data space is also modified.

3.1.2 Approximate Posterior and Evidence

The posterior and evidence of StPCA do not have closed form solutions, but these distributions are both of interest to us. We show how we obtain approximations to these, and to keep syntax clean we first introduce the parameter vector:

$$\theta := \begin{bmatrix} \mathrm{vec}(\mathbf{W}) \\ \sigma^2 \end{bmatrix} \tag{3.17}$$

which is a column vector of length $dk+1 =: n_\theta$ constructed by stacking the columns of $\mathbf{W}$ and appending σ^2.

Sine the evidence is intractable, we use the Empirical Bayes approximation (detailed in **Section 1.2.2**), approximating the evidence as the marginal likelihood with the maximum-marginal-likelihood hyperparameters:

$$p(\theta|\mathbf{X}) \approx p(\theta|\mathbf{X}, \hat{\beta}), \quad \hat{\beta} = \arg\max_{\beta} p(\mathbf{X}|\beta) \tag{3.18}$$

this avoids marginalising over the hyperparameter posterior, and instead selects just a single value of *beta*.

Neither $p(\theta|\mathbf{X}, \hat{\beta})$ nor $p(\mathbf{X}|\beta)$ are available in closed form. However, performing the Laplace approximation (**Section 1.2.1**) to the marginal likelihood gives us both of these. We take the logarithm of the un-normalised posterior, Taylor expand around the mode $\hat{\theta}$ to second order and exponentiate, producing:

$$p(\mathbf{X}|\mathbf{W}, \sigma^2)p(\mathbf{W}|\beta)p(\sigma^2) \approx p(\mathbf{X}|\hat{\mathbf{W}}, \hat{\sigma}^2)p(\hat{\mathbf{W}}|\beta)p(\hat{\sigma}^2) \exp\left\{-\frac{1}{2}(\theta - \hat{\theta})^\top \mathbf{H}_\beta (\theta - \hat{\theta})\right\} \tag{3.19}$$

which requires computing the maximum a posteriori parameters

$$\hat{\theta} := \begin{bmatrix} \mathrm{vec}(\hat{\mathbf{W}}) \\ \hat{\sigma}^2 \end{bmatrix} = \arg\max_\theta \; p(\mathbf{X}|\mathbf{W}, \sigma^2)p(\mathbf{W}|\beta)p(\sigma^2) \tag{3.20}$$

The matrix $\mathbf{H}_\beta$ is the negative Hessian of the log posterior evaluated at the MAP parameters $\hat{\theta}$. We make the evaluation at β explicit in the subscript as this will later need to be distinguished from the same matrix evaluated at $\hat{\beta}$. $\mathbf{H}_\beta$ is defined element-wise as

$$(\mathbf{H}_\beta)_{ij} := -\frac{\partial^2 \ln p(\mathbf{X}|\mathbf{W}, \sigma^2)p(\mathbf{W}|\beta)p(\sigma^2)}{\partial\theta_i \partial\theta_j}\Big|_{\theta=\hat{\theta}} \tag{3.21}$$

which is a $n_\theta \times n_\theta$ matrix. Although this is the negative Hessian of the log posterior, the intractable normalising factor $p(\mathbf{X}|\beta)$ does not contribute. We thus only require the prior and likelihood, both of which we have in closed form, enabling a closed form for **Equation 3.21**.

The time complexity of computing $|\mathbf{H}_\beta|$ is $\mathcal{O}(n_\theta^3)$ using the standard LU decomposition [Trefethen and Bau, 1997, Lecture 20]. Storage and computation may thus become challenging under large d, so we make further approximations to $\mathbf{H}_\beta$. We define $\tilde{\mathbf{H}}_\beta \approx \mathbf{H}_\beta$ as a block-diagonal approximation. $\tilde{\mathbf{H}}_\beta$ has $k+1$ blocks: k blocks of size $d \times d$ containing the covariance for each $\mathbf{w}_i$, and a single 1×1 block for the variance of σ^2. These blocks are denoted $\mathbf{H}_\beta^{\mathbf{w}_i}$ $\forall i \in 1 \cdots k$ and $\mathbf{H}_\beta^{\sigma^2}$. The form for $\tilde{\mathbf{H}}_\beta$ is discussed further in **Section 3.1.2**. Using this approximation only d^2k+1 elements need to be stored, and the determinant may be computed efficiently as $|\tilde{\mathbf{H}}_\beta| = |\mathbf{H}_\beta^{\sigma^2}| \prod_{i=1}^k |\mathbf{H}_\beta^{\mathbf{w}_i}|$. $\tilde{\mathbf{H}}_\beta^{-1}$ may also be computed by separately inverting each of the blocks.

The approximate marginal likelihood is obtained by integrating out θ from

Equation 3.19. This is a Gaussian integral, giving the closed form solution

$$p(\mathbf{X}|\beta) \approx p(\mathbf{X}|\hat{\mathbf{W}}, \hat{\sigma}^2)p(\hat{\mathbf{W}}|\beta)p(\hat{\sigma}^2)(2\pi)^{\frac{dk+1}{2}}|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}} \tag{3.22}$$

Finally, the distribution $p(\mathbf{W}, \sigma^2|\mathbf{X}, \beta)$ is related to those we have already approximated via Bayes' rule:

$$p(\mathbf{W}, \sigma^2|\mathbf{X}, \beta) = \frac{p(\mathbf{X}|\mathbf{W}, \sigma^2)p(\mathbf{W}|\beta)}{p(\mathbf{X}|\beta)} \tag{3.23}$$

which, substituting in the approximate distributions, is an un-normalised Gaussian distribution divided by its normalisation constant, giving the Gaussian approximation:

$$p(\mathbf{W}, \sigma^2|\mathbf{X}, \beta) \approx \mathcal{N}(\theta|\hat{\theta}, \tilde{\mathbf{H}}_\beta^{-1}) \tag{3.24}$$

Now that we have approximate forms for $p(\mathbf{W}, \sigma^2|\mathbf{X}, \beta)$ and $p(\mathbf{X}|\beta)$, we obtain a posterior we can compute:

$$p(\mathbf{W}, \sigma^2|\mathbf{X}) \approx \mathcal{N}(\theta|\hat{\theta}, \tilde{\mathbf{H}}_{\hat{\beta}}^{-1}) \tag{3.25}$$

this requires both $\hat{\beta}$ (appearing in $\tilde{\mathbf{H}}_{\hat{\beta}}^{-1}$) and $\hat{\theta}$.

$$\hat{\theta} = \arg\max_\theta \; p(\mathbf{X}|\mathbf{W}, \sigma^2)p(\mathbf{W}|\hat{\beta})p(\sigma^2) \tag{3.26}$$

$$\hat{\beta} = \arg\max_\beta \; p(\hat{\mathbf{W}}|\beta)|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}} \tag{3.27}$$

These two maximisations depend on each other; to solve both we may alternate between maximising θ with fixed β, and maximising β with fixed θ.

Equation 3.26 performs MAP estimation of the StPCA parameters. This may be performed efficiently using Expectation Maximisation, and is discussed in **Section 3.3.1**.

Equation 3.27 is low-dimensional so may be maximised with standard optimisation techniques. As long as the covariance function used is differentiable with respect to the hyper-parameters, analytic gradients may be obtained enabling gradient based optimisation techniques. This is shown in **Section 3.3.2**.

Structure of $\tilde{\mathbf{H}}_\beta$

As mentioned above, we use a block-diagonal approximation $\tilde{\mathbf{H}}_\beta \approx \mathbf{H}_\beta$:

$$\tilde{\mathbf{H}}_\beta := \begin{bmatrix} \mathbf{H}_\beta^{\mathbf{w}_1} & \mathbf{0} & \cdots & \cdots & \mathbf{0} \\ \mathbf{0} & \mathbf{H}_\beta^{\mathbf{w}_2} & \mathbf{0} & \cdots & \mathbf{0} \\ \vdots & \mathbf{0} & \ddots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \mathbf{H}_\beta^{\mathbf{w}_k} & \mathbf{0} \\ \mathbf{0} & \mathbf{0} & \cdots & \mathbf{0} & \mathbf{H}_\beta^{\sigma^2} \end{bmatrix} \tag{3.28}$$

where the blocks are defined as

$$\mathbf{H}_\beta^{\sigma^2} = \frac{\partial^2 -\ln p(\mathbf{X}|\mathbf{W}, \sigma^2) p(\sigma^2)}{(\partial \sigma^2)^2}\Big|_{\hat{\theta}} \tag{3.29}$$

$$\mathbf{H}_\beta^{\mathbf{w}_i} = \frac{\partial^2 -\ln p(\mathbf{X}|\mathbf{W}, \sigma^2) p(\mathbf{W}|\beta)}{\partial \mathbf{w}_i \partial \mathbf{w}_i^\top}\Big|_{\hat{\theta}} \quad \forall i \in 1 \cdots k \tag{3.30}$$

The blocks are available in closed form:

$$\mathbf{H}_\beta^{\sigma^2} = \mathrm{Tr}\Big[\mathbf{X}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{X}^\top\Big] - \frac{n}{2}\mathrm{Tr}\Big[\mathbf{C}^{-1}\mathbf{C}^{-1}\Big] - \sigma^{-4} \tag{3.31}$$

$$\begin{aligned} \mathbf{H}_\beta^{\mathbf{w}_i} =& \mathbf{K}_\beta^{-1} + \\ & \mathbf{C}^{-1}\left(\mathbf{w}_i^\top \mathbf{C}^{-1}\mathbf{X}^\top \mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i + n - n\mathbf{w}_i^\top \mathbf{C}^{-1}\mathbf{w}_i\right) + \\ & \mathbf{C}^{-1}\Big(\mathbf{X}^\top \mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i\mathbf{w}_i^\top + \mathbf{w}_i\mathbf{w}_i^\top \mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X} + \\ & \quad \left(\mathbf{w}_i^\top \mathbf{C}^{-1}\mathbf{w}_i - 1\right)\mathbf{X}^\top\mathbf{X} - n\mathbf{w}_i\mathbf{w}_i^\top\Big)\mathbf{C}^{-1} \end{aligned} \tag{3.32}$$

Detailed derivations of these are given in **Appendix D.2.1**.

Toy Example

Here we present a toy example, simulating data from an StPCA model to illustrate the type of data to which StPCA applies. We consider data which has a 1-dimensional spatial structure, which is reflected by choosing $\mathbf{t}_i = \frac{i}{d}$ and the covariance function as the noisy Squared Exponential with length scale 0.2, signal variance 1 and noise variance 10^{-5}. This small noise variance is large enough such that the matrix $\mathbf{K}$ may be inverted without numerical problems, but small enough to produce smooth loadings.

We choose a latent dimensionality of 2, observation noise variance of 0.05, and $d = 100$. The columns of $\mathbf{W}$ are plotted in **Figure 3.2**, illustrating their spatial structure. Since the synthetic data are constructed from a linear combination of

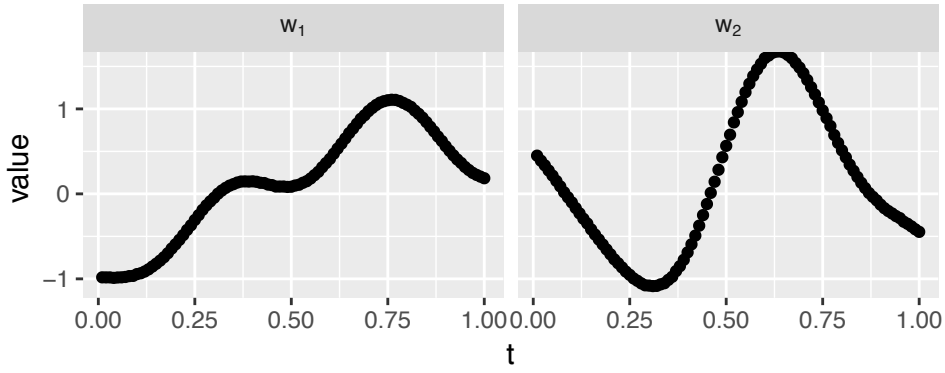


Figure 3.2: *The columns of* $\mathbf{W}$ *in the toy example described, showing their spatial structure.*

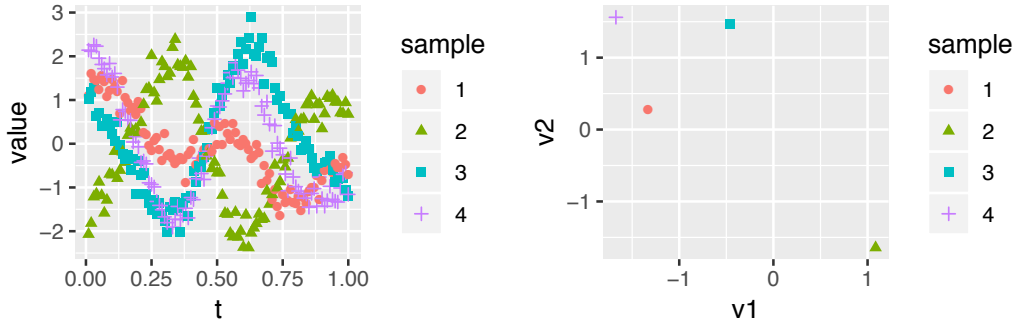


Figure 3.3: *Left: Synthetic observations constructed by taking a linear combination of the columns of* $\mathbf{W}$*, and adding white noise. Right: The latent space. One can see how a given sample* $\mathbf{x}_i$ *can be decomposed into* $\mathbf{v}_{i1}\mathbf{w_1} + \mathbf{v}_{i2}\mathbf{w_2} + noise$.

these, the synthetic data also has a spatial struture (illustrated in **Figure 3.3**).

Impact of the Laplace approximation

The Laplace approximation is used to produce an approximate posterior and marginal likelihood. Whilst this is an improvement upon using a point estimate in terms of quantifying uncertainty (assuming appropriateness of the approximation), the Laplace approximation is a strong assumption. Specifically the assumption is that the entire posterior is unimodal and well represented by a Gaussian distribution fit to the local curvature around the mode. We do not have any guarantees that this is true, however we do find good empirical performance in **Chapter 4**.

Using the Laplace approximation, the posterior is reported as the location of a mode and covariance matrix of an approximating Gaussian, and the marginal likelihood is obtained as the normalising constant of this Gaussian. If the true posterior is not well approximated, then the MAP will still be correct, but the

covariance matrix may not well describe the true posterior covariance structure. Also, if the approximation is poor, then the approximate marginal likelihood may be inaccurate causing poor hyper-parameter values to be selected.

An alternative to the Laplace approximation would be to take an MCMC approach and sample directly from the posterior, which has the property of giving the asymptotically correct posterior given enough computation. In modern probabilistic programming languages such as Stan [Carpenter et al., 2017] this should be relatively easy to implement given the model defined in **Equation 3.5** – **Equation 3.9**. One would need to place proper priors on β and σ^2 to make the model truly generative. Care must also be taken with sampling values of β; for each new sample the $\mathcal{O}(d^3)$ operation of decomposing $\mathbf{K}_\beta$ is required (so that values can be drawn from $p(\mathbf{W}|\beta)$). This is an expensive operation so it may be preferable to limit β to a small number of values and optimise.

3.2 The covariance structure prior $p(\mathbf{W}|\beta)$

The prior $p(\mathbf{W}|\beta)$ is used to incorporate additional information into learning the loadings matrix $\mathbf{W}$. Since $\mathbf{W}$ is used to describe the covariance structure of the data (**Equation 3.15**), $p(\mathbf{W}|\beta)$ encodes prior knowledge of the covariance structure of the data.

The prior has the form

$$p(\mathbf{W}|\beta) = \prod_{i=1}^{k} p(\mathbf{w}_i|\beta) \tag{3.33}$$

$$= \prod_{i=1}^{k} \mathcal{N}(\mathbf{w}_i|\mathbf{0}, \mathbf{K}_\beta) \tag{3.34}$$

which decomposes into k independent identical Gaussian priors, each over a column of the loadings.

Each of the d input variables, and thus each of the elements of each $\mathbf{w}_i$, is associated with a location vector $\mathbf{t}_i$. In this thesis we focus on the case where each of the features of the input corresponds to a pixel in a 2d image, and the corresponding $\mathbf{t}_i$ is a length 2 column vector encoding the coordinates of the pixel. We note that this model can easily be applied to a wide range of other location types such as 1d time-series or 3d images. In principle one could even use strings or other structured objects instead of real-value vectors by using a covariance function defined on the appropriate domain [e.g., Rasmussen and Williams, 2005, Section 4.4], but we do

not explore this.

The prior over $\mathbf{w}_i$ is mean $\mathbf{0}$ with a covariance matrix constructed element-wise as

$$(\mathbf{K}_\beta)_{ij} = C(\mathbf{t}_i, \mathbf{t}_j; \beta) \quad \forall\, i, j \in 1 \dots d \tag{3.35}$$

where $C(\,\cdot\,, \cdot\,; \beta)$ is a covariance function parametrised by the hyper-parameters β. This is a Gaussian process prior over $\mathbf{W}$. There is a large body of literature on covariance functions, and we summarise the relevant material in **Section 1.3.2**.

A covariance function is an intuitive way to specify a prior over high dimensional covariance structure. A number of well-known covariance functions are interpretable, with interpretable hyper-parameters. For example, selecting the squared exponential covariance function encodes locality and smoothness, whilst the exponential covariance function similarly encodes locality, but without smoothness [e.g., Rasmussen and Williams, 2005, Chapter 4].

If d is large, storing the $d \times d$ covariance matrix $\mathbf{K}_\beta$ may become computationally challenging, as will the $\mathcal{O}(d^3)$ inversion required to compute the value of the prior. To facilitate large d, one can use *compactly supported covariance functions*; stationary covariance functions which equal identically zero for large separation between inputs [Wendland, 1995; Melkumyan and Ramos, 2009].

3.2.1 Prior data covariance

In StPCA we place a prior over the loadings matrix $\mathbf{W}$ to express our prior knowledge of the covariance structure of the data. The form of the prior data distribution, $p(\mathbf{x}|\beta)$, would show how the prior over the loadings affects our prior over the data, if at all. However, this distribution is improper and analytically difficult to obtain due to the improper prior in σ^2. For this reason we study $p(\mathbf{x}|\sigma^2, \beta)$; our prior expectation of how the data should be distributed given that we know the magnitude of the noise.

$p(\mathbf{x}|\beta, \sigma^2)$ does not have a closed form solution. However, we are primarily interested in the covariance, since it is covariance structure we are imposing through $p(\mathbf{W}|\beta)$. Computing moments is tractable, so we study the covariance of $p(\mathbf{x}|\beta, \sigma^2)$.

The mean of $p(\mathbf{x}|\beta, \sigma^2)$ is $\mathbf{0}$, so the covariance is calculated as

$$\mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]_{p(\mathbf{x}|\beta,\sigma^2)} = \int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^\top p(\mathbf{x}|\beta, \sigma^2)\, \mathrm{d}\mathbf{x} \tag{3.36}$$

$$= \int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^\top \int_{\mathbb{R}^{d\times k}} p(\mathbf{x}|\mathbf{W}, \sigma^2) p(\mathbf{W}|\beta)\, \mathrm{d}\mathbf{W}\, \mathrm{d}\mathbf{x} \tag{3.37}$$

$$= \int_{\mathbb{R}^{d\times k}} \left[\int_{\mathbb{R}^d} \mathbf{x}\mathbf{x}^\top p(\mathbf{x}|\mathbf{W}, \sigma^2)\, \mathrm{d}\mathbf{x}\right] p(\mathbf{W}|\beta)\, \mathrm{d}\mathbf{W} \tag{3.38}$$

$$= \int_{\mathbb{R}^{d\times k}} \mathbb{E}\left[\mathbf{x}\mathbf{x}^\top\right]_{p(\mathbf{x}|\mathbf{W},\sigma^2)} p(\mathbf{W}|\beta)\, \mathrm{d}\mathbf{W} \tag{3.39}$$

$$= \int_{\mathbb{R}^{d\times k}} (\mathbf{W}\mathbf{W}^\top + \sigma^2 I) p(\mathbf{W}|\beta)\, \mathrm{d}\mathbf{W} \tag{3.40}$$

$$= \mathbb{E}\left[\mathbf{W}\mathbf{W}^\top\right]_{p(\mathbf{W}|\beta)} + \sigma^2 I \tag{3.41}$$

$$= \sum_{i=1}^{k} \mathbb{E}\left[\mathbf{w}_i \mathbf{w}_i^\top\right]_{p(\mathbf{w}_i|\beta)} + \sigma^2 I \tag{3.42}$$

$$= k\mathbf{K}_\beta + \sigma^2 I \tag{3.43}$$

$p(\mathbf{x}|\beta, \sigma^2)$ thus has the same covariance as $p(\mathbf{w}_i|\beta)$, but scaled by k and corrupted by white noise.

3.3 Inference

To compute the approximate posterior, we need to find $\hat{\theta}$ and $\hat{\beta}$ (**Equations 3.27 and 3.26**). We derive these procedures here.

3.3.1 Inferring θ

The maximum-a-posteriori parameters for StPCA, $\hat{\theta}$, are not available in closed form. However, they may be computed using Expectation Maximisation (EM), which we derive below. This is similar to the EM procedure in PPCA [Tipping and Bishop, 1999], but the prior over $\mathbf{W}$ adds some complications which will be seen.

For the expectation step, we compute the expected complete log posterior with respect to the posterior over the latent variables. Since we are maximising this,

we are free to drop terms which are constant in θ:

$$\mathbb{E}\left[\ln p(\mathbf{W}, \sigma^2 | \mathbf{X}, \mathbf{V}, \beta)\right] \tag{3.44}$$

$$\propto \mathbb{E}\left[\ln p(\mathbf{X}, \mathbf{V} | \mathbf{W}, \sigma^2) + \ln p(\mathbf{W}|\beta) + \ln p(\sigma^2)\right] \tag{3.45}$$

$$= \mathbb{E}\left[\ln p(\mathbf{X}|\mathbf{V}, \mathbf{W}, \sigma^2)\right] + \mathbb{E}\left[\ln p(\mathbf{V})\right] + \ln p(\mathbf{W}|\beta) + \ln p(\sigma^2) \tag{3.46}$$

$$\begin{aligned} = & -\frac{1}{2}\sum_{i=1}^{n}\Bigg\{ d\ln 2\pi\sigma^2 + \mathrm{Tr}\left[\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]\right] + \sigma^{-2}\mathbf{x}_i^\top\mathbf{x}_i \\ & \qquad - 2\sigma^{-2}\mathbb{E}\left[\mathbf{v}_i\right]^\top\mathbf{W}^\top\mathbf{x}_i + \sigma^{-2}\,\mathrm{Tr}\left[\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]\mathbf{W}^\top\mathbf{W}\right] \\ & \qquad + k\ln 2\pi\Bigg\} \\ & -\frac{1}{2}\sum_{i=1}^{k}\left\{ d\ln 2\pi + \ln|\mathbf{K}_\beta| + \mathbf{w}_i^\top\mathbf{K}_\beta^{-1}\mathbf{w}_i\right\} - \ln\sigma^2 \end{aligned} \tag{3.47}$$

We can see that the expected complete log likelihood depends only on the expected first and second order statistics $\mathbb{E}\left[\mathbf{v}_i\right]$, $\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]$; the sufficient statistics for the Gaussian. Note that, unless specified with a subscript, all expectations are taken with respect to the posterior over the latent variables given the old parameter values $p(\mathbf{V}|\mathbf{X}, \mathbf{W}_{\text{old}}, \sigma^2_{\text{old}}, \beta_{\text{old}})$. In the expectation step of EM we only need to compute these sufficient statistics:

$$\mathbb{E}\left[\mathbf{v}_i\right] = \left(\mathbf{W}^\top\mathbf{W} + \sigma^2 I\right)^{-1}\mathbf{W}^\top\mathbf{x}_i \tag{3.48}$$

$$\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right] = \sigma^2\left(\mathbf{W}^\top\mathbf{W} + \sigma^2 I\right)^{-1} + \mathbb{E}\left[\mathbf{v}_i\right]\mathbb{E}\left[\mathbf{v}_i\right]^\top \tag{3.49}$$

The maximisation stage of EM is more complicated, as we are unable to obtain a closed form expression for the parameters jointly maximising

$$(\mathbf{W}_{\text{new}}, \sigma^2_{\text{new}}) = \underset{\mathbf{W},\sigma^2}{\arg\max}\ \mathbb{E}\left[\ln p(\mathbf{W}, \sigma^2|\mathbf{X}, \mathbf{V}, \beta)\right] \tag{3.50}$$

However, we can obtain expressions for maximising $p(\mathbf{W}, \sigma^2|\mathbf{X}, \mathbf{V})$ for $\mathbf{W}$ with fixed σ^2, and for σ^2 with fixed $\mathbf{W}$. One could iterate between these until convergence; however, performing just a single iteration of each is sufficient. For an EM procedure to converge, it is only required that, after the M-step,

$$\mathbb{E}\left[\ln p(\mathbf{W}_{\text{new}}, \sigma^2_{\text{new}}|\mathbf{X}, \mathbf{V}, \beta)\right] \geq \mathbb{E}\left[\ln p(\mathbf{W}_{\text{old}}, \sigma^2_{\text{old}}|\mathbf{X}, \mathbf{V}, \beta)\right] \tag{3.51}$$

This is guaranteed by performing maximisation with respect to σ^2 and then with respect to $\mathbf{W}$. These maximisation steps are derived below. This incomplete max-

imisation is an instance of generalised EM [Murphy, 2012, Section 11.4.9].

The M-step for σ^2 is relatively simple:

$$0 = \frac{\partial}{\partial \sigma^2_{\text{new}}} \mathbb{E}\left[\ln p(\mathbf{W}, \sigma^2_{\text{new}} | \mathbf{X}, \mathbf{V}, \beta)\right] \tag{3.52}$$

$$= -\left(\frac{nd}{2} + 1\right) \frac{\partial \ln \sigma^2_{\text{new}}}{\partial \sigma^2_{\text{new}}} - \frac{1}{2} \frac{\partial \sigma^{-2}_{\text{new}}}{\partial \sigma^2_{\text{new}}} \sum_{i=1}^{n} \left\{ \mathbf{x}_i^\top \mathbf{x}_i - 2\mathbb{E}\left[\mathbf{v}_i\right]^\top \mathbf{W}^\top \mathbf{x}_i + \text{Tr}\left[\mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \mathbf{W}^\top \mathbf{W}\right] \right\} \tag{3.53}$$

Solving the partial derivatives and rearranging, gives us

$$\sigma^2_{\text{new}} = \frac{1}{nd + 2} \sum_{i=1}^{n} \left\{ \mathbf{x}_i^\top \mathbf{x}_i - 2\mathbb{E}\left[\mathbf{v}_i\right]^\top \mathbf{W}^\top \mathbf{x}_i + \text{Tr}\left[\mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \mathbf{W}^\top \mathbf{W}\right] \right\} \tag{3.54}$$

this is no different to the maximisation step for σ^2 in PPCA, which is to be expected since $p(\mathbf{W}|\mathbf{K}_\beta)$ does not depend on σ^2.

Maximising $\mathbf{W}$ is more complicated. Here we again take partial derivatives and attempt to solve for $\mathbf{W}$.

$$\mathbf{0} = \frac{\partial}{\partial \mathbf{W}} \mathbf{E}\left[\ln p(\mathbf{W}, \sigma^2 | \mathbf{X}, \mathbf{V}, \beta)\right] \tag{3.55}$$

$$= -\frac{1}{2} \sum_{i=1}^{n} \left\{ -2\sigma^{-2} \mathbb{E}\left[\mathbf{v}_i\right]^\top \frac{\partial \mathbf{W}^\top}{\partial \mathbf{W}} \mathbf{x}_i + 2\sigma^{-2} \frac{\partial}{\partial \mathbf{W}} \text{Tr}\left[\mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \mathbf{W}^\top \mathbf{W}\right] \right\} - \frac{1}{2} \sum_{i=1}^{k} \left\{ \frac{\partial \mathbf{w}_i^\top \mathbf{K}_\beta^{-1} \mathbf{w}_i}{\partial \mathbf{W}} \right\} \tag{3.56}$$

$$= \sum_{i=1}^{n} \left\{ \sigma^{-2} \mathbb{E}\left[\mathbf{v}_i\right]^\top \mathbf{x}_i - \sigma^{-2} \mathbf{W} \mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \right\} - \frac{1}{2} \frac{\partial}{\partial \mathbf{W}} \text{Tr}\left[\mathbf{W}^\top \mathbf{K}_\beta \mathbf{W}\right] \tag{3.57}$$

$$= \left[\sum_{i=1}^{n} \mathbb{E}\left[\mathbf{v}_i\right]^\top \mathbf{x}_i \right] - \mathbf{W} \left[\sum_{i=1}^{n} \mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \right] - \sigma^2 \mathbf{K}_\beta^{-1} \mathbf{W} \tag{3.58}$$

$\implies$

$$\left[\sigma^{-2} \mathbf{K}_\beta\right] \mathbf{W} + \mathbf{W} \left[\sum_{i=1}^{n} \mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \right]^{-1} = \left[\sigma^{-2} \mathbf{K}_\beta\right] \left[\sum_{i=1}^{n} \mathbb{E}\left[\mathbf{v}_i\right]^\top \mathbf{x}_i \right] \left[\sum_{i=1}^{n} \mathbb{E}\left[\mathbf{v}_i \mathbf{v}_i^\top\right] \right]^{-1} \tag{3.59}$$

$=:$

$$\mathbf{A}\mathbf{W} + \mathbf{W}\mathbf{B} = \mathbf{C} \tag{3.60}$$

This is a Sylvester equation for $\mathbf{W}$ [e.g., Bhatia and Rosenthal, 1997]. In this particular case we have positive-semidefinite $\mathbf{A}$ and positive-definite $\mathbf{B}$, making an

analytic solution tractable. The positive-definiteness for $\mathbf{B}$ can be seen using the definition of $\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]$ in **Equation 3.49** to expand out

$$\sum_{i=1}^{n}\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right] = n\sigma^2\mathbf{M}^{-1} + \left(\mathbf{X}\mathbf{W}\mathbf{M}^{-1}\right)^\top\mathbf{X}\mathbf{W}\mathbf{M}^{-1} \tag{3.61}$$

where $\mathbf{M} = \mathbf{W}^\top\mathbf{W} + \sigma^2 I$. The first term is a positive multiple of the positive-definite $\mathbf{M}^{-1}$, and the second term is positive-semidefinite. The sum of the two thus produces a positive definite $\mathbf{B}$.

Below we show how to solve for $\mathbf{W}$. Taking **Equation 3.60**, we eigen-decompose $\mathbf{B} = \mathbf{Q}_\mathbf{B}\mathbf{\Lambda}_\mathbf{B}\mathbf{Q}_\mathbf{B}^\top$ and post-multiply by $\mathbf{Q}_\mathbf{B}$:

$$\begin{aligned}
\mathbf{A}\mathbf{W} + \mathbf{W}\mathbf{B} &= \mathbf{C} && (3.62)\\
\mathbf{A}\mathbf{W} + \mathbf{W}\mathbf{Q}_\mathbf{B}\mathbf{\Lambda}_\mathbf{B}\mathbf{Q}_\mathbf{B}^\top &= \mathbf{C} && (3.63)\\
\mathbf{A}\mathbf{W}\mathbf{Q}_\mathbf{B} + \mathbf{W}\mathbf{Q}_\mathbf{B}\mathbf{\Lambda}_\mathbf{B} &= \mathbf{C}\mathbf{Q}_\mathbf{B} && (3.64)\\
\mathbf{A}\tilde{\mathbf{W}} + \tilde{\mathbf{W}}\mathbf{\Lambda}_\mathbf{B} &= \tilde{\mathbf{C}} && (3.65)
\end{aligned}$$

Now, looking at individual columns of $\tilde{\mathbf{C}}$,

$$\begin{aligned}
\tilde{\mathbf{C}}_i &= (\mathbf{A}\tilde{\mathbf{W}})_i + (\tilde{\mathbf{W}}\mathbf{\Lambda}_\mathbf{B})_i && (3.66)\\
&= (\mathbf{A} + \lambda_i I)\tilde{\mathbf{W}}_i && (3.67)
\end{aligned}$$

The matrix $(\mathbf{A}+\lambda_i I)$ is always positive definite due to adding $\lambda_i > 0$ to the diagonal of $\mathbf{A}$. We can thus invert this and solve for $\tilde{\mathbf{W}}$.

$$\begin{aligned}
\tilde{\mathbf{W}}_i &= (\mathbf{A} + \lambda_i I)^{-1}\tilde{\mathbf{C}}_i && (3.68)\\
&= \left(\sigma^{-2}\mathbf{K}_\beta + \lambda_i I\right)^{-1}\left(\left[\sigma^{-2}\mathbf{K}_\beta\right]\left[\sum_{i=1}^{n}\mathbb{E}\left[\mathbf{v}_i\right]^\top\mathbf{x}_i\right]\mathbf{Q}_\mathbf{B}\mathbf{\Lambda}_\mathbf{B}\right)_i && (3.69)
\end{aligned}$$

So $\tilde{\mathbf{W}}$ may be calculated column-wise, and $\mathbf{W}$ can be recovered as

$$\mathbf{W} = \tilde{\mathbf{W}}\mathbf{Q}_\mathbf{B}^\top \tag{3.70}$$

Implementation notes

This computation can be made efficient. $\mathbf{B}^{-1} = \sum_{i=1}^{n}\mathbb{E}\left[\mathbf{v}_i\mathbf{v}_i^\top\right]$ is constructed as part of the EM procedure in θ inference; we can obtain $\mathbf{Q}_\mathbf{B}$ by eigen-decomposing $\mathbf{B}^{-1}$ without any inversion.

We do need to solve the large linear system $(\mathbf{A} + \lambda_i I)^{-1}\tilde{\mathbf{C}}_i$. However, if a

compactly supported covariance function is used, $\mathbf{A} = \sigma^{-2}\mathbf{K}_\beta$ is sparse, and adding σ^2 to the diagonal does not change the sparsity pattern, so the linear system may be solved efficiently. This is taken advantage of using a sparse solver in the `stpca` package.

The number of EM iterations required for convergence is reduced if we start off with a good guess at the MAP values. Making this guess must be computationally cheap otherwise we could have simply run the EM algorithm for longer. In `stpca` we initialise σ^2 and $\mathbf{W}$ from the closed from maximum likelihood parameters for PPCA as per **Equation 1.18** and **Equation 1.19**. Convergence has empirically been found to be much faster using PPCA initialisation over random initialisation.

3.3.2 Inferring β

Inference of β requires performing the maximisation in **Equation 3.27**, which is repeated here:

$$\hat{\beta} = \arg\max_{\beta} p(\hat{\mathbf{W}}|\beta)|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}} \tag{3.71}$$

This maximisation is performed in StPCA through the iterative BFGS method [Fletcher, 1987, Algorithms 2.6.2 and 2.6.4]. BFGS was chosen since it takes advantage of function gradients, which are desirable since the optimisation is low dimension (the number of hyper-parameters) and gradients are analytic in this case.

We focus on maximising the logarithm of $p(\hat{\mathbf{W}}|\beta)|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}}$, which is analytically simpler and more computationally stable, since probability densities can get small enough for numerical precision to be important. We thus require the partial derivatives $\frac{\partial \log p(\hat{\mathbf{W}}|\beta)|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}}}{\partial \beta_i}$ for each $\beta_i \in \beta$.

The covariance function used to construct $\mathbf{K}_\beta$ is specified by the user of StPCA. We assume here that the user also supplies a function giving the partial derivatives of the covariance function with respect to the hyper-parameters:

$$\frac{\partial C(\mathbf{t}_j, \mathbf{t}_k; \beta)}{\partial \beta_i} = \left(\frac{\partial \mathbf{K}_\beta}{\partial \beta_i}\right)_{jk} \tag{3.72}$$

Not all covariance functions admit an analytic derivative, but in these cases a numerical derivative may still be used. This allows us to obtain expressions for

$$\frac{\partial \log p(\hat{\mathbf{W}}|\beta)|\tilde{\mathbf{H}}_\beta|^{-\frac{1}{2}}}{\partial \beta_i} = \frac{\partial}{\partial \beta_i}\left[\log p(\hat{\mathbf{W}}|\beta) - \frac{1}{2}\log|\tilde{\mathbf{H}}_\beta|\right] \tag{test}$$

First we expand out the definition of $\log p(\hat{\mathbf{W}}|\beta)$ (**Equation ??**), dropping constant terms. We also recall that $|\tilde{\mathbf{H}}_\beta|$ is block-diagonal, so its log determinant is the sum

of the log determinants of each block, and we can drop the $\mathbf{H}_\beta^{\sigma^2}$ since it is constant with respect to β. This gives:

$$-\frac{1}{2}\frac{\partial}{\partial\beta_i}\left[k\log|\mathbf{K}_\beta| + \mathrm{Tr}\left[\hat{\mathbf{W}}^\top\mathbf{K}_\beta^{-1}\hat{\mathbf{W}}\right] + \sum_{j=1}^{k}\log|\mathbf{H}_\beta^{\mathbf{w}_j}|\right] \tag{3.73}$$

Now we apply the derivative operator to each of the three terms. This uses the identity for the derivative of a matrix determinant (**Equation A.8**) for the first and third terms. For the second term, the step can be performed by expanding out $\mathrm{Tr}\left[\hat{\mathbf{W}}^\top\mathbf{K}_\beta^{-1}\hat{\mathbf{W}}\right] = \sum_{i=1}^{k}\hat{\mathbf{w}}_i^\top\mathbf{K}_\beta^{-1}\hat{\mathbf{w}}_i$, moving the derivative operator inside and using the identity for the derivative of a matrix inverse (**Equation A.9**). This arrives at:

$$-\frac{1}{2}\left[k\,\mathrm{Tr}\left[\mathbf{K}_\beta^{-1}\frac{\partial\mathbf{K}_\beta}{\partial\beta_i}\right] - \mathrm{Tr}\left[\hat{\mathbf{W}}^\top\mathbf{K}_\beta^{-1}\frac{\partial\mathbf{K}_\beta}{\partial\beta_i}\mathbf{K}_\beta^{-1}\hat{\mathbf{W}}\right] - \sum_{j=1}^{k}\mathrm{Tr}\left[\mathbf{H}_\beta^{\mathbf{w}_j}\mathbf{K}_\beta^{-1}\frac{\partial\mathbf{K}_\beta}{\partial\beta_i}\mathbf{K}_\beta^{-1}\right]\right] \tag{3.74}$$

Finally we re-arrange for readability:

$$-\frac{1}{2}\,\mathrm{Tr}\left[\left(k\mathbf{K}_\beta^{-1} - \mathbf{K}_\beta^{-1}\hat{\mathbf{W}}\hat{\mathbf{W}}^\top\mathbf{K}_\beta^{-1} - \sum_{j=1}^{k}\mathbf{K}_\beta^{-1}\mathbf{H}_\beta^{\mathbf{w}_j}\mathbf{K}_\beta^{-1}\right)\frac{\partial\mathbf{K}_\beta}{\partial\beta_i}\right] \tag{3.75}$$

Computational considerations

The bulk of the computation required in StPCA is in the β-tuning stage due to requiring $\mathbf{K}_\beta^{-1}$ in $p(\mathbf{W}|\beta)$ as well as each $\mathbf{H}_\beta^{\mathbf{w}_i}$. In `stpca` we perform a single decomposition of $\mathbf{K}_\beta$ and use this whenever required, but this is still a computationally heavy operation, requiring $\mathcal{O}(d^3)$ operations every β-tuning iteration. When using a compactly supported covariance function, a computationally cheaper sparse matrix decomposition is performed. However, if the feature covariance length scales are long, $\mathbf{K}_\beta$ will remain dense.

To scale StPCA to higher dimensional data, it would be possible to take advantage of reduced rank covariance approximations which are common in the Gaussian Processes literature [e.g., Rasmussen and Williams, 2005, Section 8.1]. Here we assume we have used a 'noisy' covariance function, so $\mathbf{K}_\beta$ has the form $\mathbf{K}_\beta^{\text{noiseless}} + \sigma^2 I$, and the noiseless part of the covariance matrix is associated with a noiseless part of the covariance function $C^{\text{noiseless}}$. If the noiseless covariance matrix is then approximated as rank m:

$$\mathbf{K}_\beta^{\text{noiseless}} \approx \mathbf{L}\mathbf{L}^\top, \quad \mathbf{L} \in \mathbb{R}^{d\times m} \tag{3.76}$$

then the calculation for the approximate $\mathbf{K}_\beta^{-1}$ can be rearranged using the matrix inversion lemma as

$$(\mathbf{L}\mathbf{L}^\top + \sigma^2 I)^{-1} = \sigma^{-2} I - \sigma^{-2}\mathbf{L}(\mathbf{L}^\top\mathbf{L} + \sigma^2 I)^{-1}\mathbf{L} \tag{3.77}$$

This form only requires inverting an $m \times m$ matrix, saving computation.

There are a number of reduced rank approximations to choose from. An approximate covariance can be constructed using only m pseudo-features with locations $\hat{\mathbf{t}}_i$, $i = 1 \ldots m$. The low-rank approximation to $\mathbf{K}_\beta^{\text{noiseless}}$ is then

$$\mathbf{K}_\beta^{\text{noiseless}} \approx \mathbf{K}_{\mathbf{t}\hat{\mathbf{t}}}\mathbf{K}_{\hat{\mathbf{t}}\hat{\mathbf{t}}}^{-1}\mathbf{K}_{\hat{\mathbf{t}}\mathbf{t}} \tag{3.78}$$

Here $\mathbf{K}_{\mathbf{t}\hat{\mathbf{t}}} = \mathbf{K}_{\hat{\mathbf{t}}\mathbf{t}}^\top$ is the $d \times m$ noiseless covariance between the real and pseudo-features such that $(\mathbf{K}_{\mathbf{t}\hat{\mathbf{t}}})_{ij} = C^{\text{noiseless}}(\mathbf{t}_i, \hat{\mathbf{t}}_j \ ; \ \beta)$. $\mathbf{K}_{\hat{\mathbf{t}}\hat{\mathbf{t}}}$ is the noiseless covariance within the pseudo-features: $(\mathbf{K}_{\hat{\mathbf{t}}\hat{\mathbf{t}}})_{ij} = C^{\text{noiseless}}(\hat{\mathbf{t}}_i, \hat{\mathbf{t}}_j \ ; \ \beta)$. Another option would be to obtain $\mathbf{L}$ as an incomplete Cholesky decomposition of $\mathbf{K}_\beta^{\text{noiseless}}$ [Fine and Scheinberg, 2001].

Initialising β

From **Equation 3.43**, we know that StPCA has a prior expectation of the covariance structure of the data which depends on $\mathbf{K}_\beta$.

$$\text{cov}(\mathbf{x}|\sigma^2, \beta) = k\mathbf{K}_\beta + \sigma^2 I \tag{3.79}$$

If we assume that the data actually have this covariance (i.e., the prior was good), one can make the equality

$$\mathbf{S} = k\mathbf{K}_\beta + \sigma^2 I \tag{3.80}$$

where $\mathbf{S} = \frac{1}{n}\mathbf{X}^\top\mathbf{X}$. This can be rearranged to give

$$\mathbf{K}_\beta = \frac{1}{k}\left(\mathbf{S} - \sigma^2 I\right) \tag{3.81}$$

which is the starting point for initialising β; we would like a value of β such that $\mathbf{K}_\beta \approx \frac{1}{k}\left(\mathbf{S} - \sigma^2 I\right)$.

Many covariance functions are of the form $C(\,\cdot\,,\cdot\,;\beta) = \sigma_f^2\tilde{C}(\,\cdot\,,\cdot\,;\beta)$, where σ_f^2 is the signal variance, and $\tilde{C}$ has the property that $\tilde{C}(\mathbf{t}, \mathbf{t}; \beta) = 1$. For such covariance functions, each element of the diagonal of **Equation 3.81** has the form

$$\sigma_f^2 = \frac{1}{k}\left(\mathbf{S}_{ii} - \sigma^2\right) \tag{3.82}$$

for each $i \in 1 \cdots d$. We thus use the following initialisation for σ_f^2:

$$\sigma_f^2 \leftarrow \frac{1}{k}\text{mean}\left(\text{diag}\left(\mathbf{S} - \sigma^2 I\right)\right) \tag{3.83}$$

Similarly, if $\tilde{C}$ is the squared exponential covariance function, a single non-diagonal element of **Equation 3.81** can be considered and rearranged to have the form

$$\ell = \sqrt{\frac{||\mathbf{t}_i - \mathbf{t}_j||_2^2}{2(\ln(\sigma_f^2 k) - \ln(\mathbf{S}_{ij}))}} \tag{3.84}$$

The reason we only consider off-diagonal elements $i \neq j$ is that the diagonal elements contain no information relevant to the length scale. We thus propose the initialiser for the length scale

$$\ell \leftarrow \text{mean}\left(\sqrt{\frac{||\mathbf{t}_i - \mathbf{t}_j||_2^2}{2(\ln(\sigma_f^2 k) - \ln(\mathbf{S}_{ij}))}} \quad \forall\, i \neq j\right) \tag{3.85}$$

3.4 Relationship to PCA

The approximate posterior reported by StPCA is centred at the MAP, which is closely related to PCA. In the case of zero noise and an infinitely broad prior, the MAP $\mathbf{W}$ is exactly equal to the principal components (PCs) scaled by the square root of their associated eigenvalues (show in **Section 3.4.2**). From this point of view, StPCA is simply PPCA with the addition of prior knowledge, and a covariance matrix describing the uncertainty around the estimated PCs as an additional output.

3.4.1 The Structured Components

In **Section 3.1**, we discussed how StPCA has a non-identifiability, where $\mathbf{W}$ may be transformed to $\mathbf{W} \rightarrow \mathbf{WR}$ for any $k \times k$ orthonormal matrix $\mathbf{R}$, as long as the latent space is similarly transformed. This non-identifiability means that once a MAP $\mathbf{W}$ has been found, transforming $\mathbf{W}$ with any $\mathbf{R}$ will also produce a value maximising the posterior.

In StPCA, we choose to apply an orthogonal transformation to the MAP $\mathbf{W}$ such that the columns of $\mathbf{W}$ are orthogonal and aligned with the eigenvectors of the likelihood covariance $\mathbf{W}\mathbf{W}^\top + \sigma^2 I$. We do this by performing the SVD of $\mathbf{W}$, and reconstructing it with the right singular vectors replaced with the $k \times k$ identity matrix.

This transformation provides additional interpretability of $\mathbf{W}$. The first

column of $\mathbf{W}$, $\mathbf{w}_1$, now has the direction of the greatest modelled variance in the data and is orthogonal to all other columns. We call the direction of these transformed columns of $\mathbf{W}$ the Structured Components (StCs) due to their similarity to the PCs in PCA; $\mathbf{w}_i/||\mathbf{w}_i||$ is the i'th StC. Note that the direction of greatest modelled variance may not be the same as the direction of greatest variance in the data, but these become equivalent in the limit of an infinitely broad $p(\mathbf{W}|\beta)$, where StPCA recovers PPCA and the StCs align with the PCs.

3.4.2 Recovering PCA from the StPCA MAP

In the limiting case of using an infinitely broad prior, StPCA becomes equal to PPCA, and the MAP parameters equal the maximum likelihood parameters. We know from PPCA that a closed form solution exists for $\mathbf{W}$ in this case [Tipping and Bishop, 1999, Appendix A], which has an interesting form to study:

$$\mathbf{W}_{\mathrm{ML}} = \arg\max_{\mathbf{W}} \log p(\mathbf{X}|\mathbf{W}, \sigma^2) \tag{3.86}$$

$$= \mathbf{Q}_k \left(\mathbf{\Lambda}_k - \sigma^2 I\right)^{\frac{1}{2}} \mathbf{R} \tag{3.87}$$

Here, $\mathbf{Q}\mathbf{\Lambda}\mathbf{Q}^\top$ is the eigen-decomposition of the sample covariance matrix $\frac{1}{n}\mathbf{X}^\top\mathbf{X}$. $\mathbf{Q}_k$ is the $d \times k$ matrix containing the first k PCs in columns, and $\mathbf{\Lambda}$ is the $k \times k$ diagonal matrix containing the corresponding eigenvalues. $\mathbf{R}$ is any orthonormal $k \times k$ matrix and is discussed in **Section 3.1**, but represents a non-identifiability in the relating to rotations in the latent space, and may be chosen to be the identity matrix.

Choosing $\mathbf{R} = I$ and $\sigma^2 = \sigma^2_{\mathrm{PCA}} = 0$, we obtain $\mathbf{W}_{\mathrm{PCA}} = \mathbf{Q}_k\mathbf{\Lambda}^{\frac{1}{2}}$ which is simply the first k PCs, each with magnitude of the square root of its corresponding eigenvalue. This produces a degenerate probability model since the likelihood covariance matrix $(\mathbf{W}\mathbf{W}^\top + \sigma^2 I)$ is low rank. However, this model exactly recovers PCA. This can be seen by plugging these values of $\mathbf{R}$ and σ^2 in to the latent-to-observed and observed-to-latent mappings in StPCA (**Equations 3.5** and **3.16** respectively):

$$p(\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}_{\mathrm{PCA}}, \sigma^2_{\mathrm{PCA}}) = \mathcal{N}(\mathbf{x}_i|\mathbf{Q}_k\mathbf{\Lambda}_k^{\frac{1}{2}}\mathbf{v}_i, \mathbf{0}) \tag{3.88}$$

$$p(\mathbf{v}_i|\mathbf{x}_i, \mathbf{W}_{\mathrm{PCA}}, \sigma^2_{\mathrm{PCA}}) = \mathcal{N}(\mathbf{v}_i|\mathbf{\Lambda}_k^{-\frac{1}{2}}\mathbf{Q}_k^\top\mathbf{x}_i, \mathbf{0}) \tag{3.89}$$

Here the covariance matrices are both zero, so we do not have probabilistic statements but deterministic mappings. These may be given in matrix form to show the mapping to the latent space and the corresponding reconstruction for the entire

dataset:

$$\mathbf{V} = \mathbf{X}(\mathbf{Q}_k \boldsymbol{\Lambda}_k^{-\frac{1}{2}}) = \mathbf{X}\mathbf{W}_{\mathrm{PCA}}^{+\top} \tag{3.90}$$

$$\mathbf{X}^{\mathrm{rec}} = \mathbf{V}(\boldsymbol{\Lambda}_k^{\frac{1}{2}} \mathbf{Q}_k^{\top}) = \mathbf{V}\mathbf{W}_{\mathrm{PCA}}^{\top} \tag{3.91}$$

$$\tag{3.92}$$

where $\mathbf{W}_{\mathrm{PCA}}^{+} = \boldsymbol{\Lambda}_k^{-\frac{1}{2}} \mathbf{Q}_k^{\top}$ is the left inverse of $\mathbf{W}_{\mathrm{PCA}}$, and $\mathbf{X}^{\mathrm{rec}}$ is the reconstruction of $\mathbf{X}$ from the latent variables $\mathbf{V}$. Comparing this to the mappings between latent and observed space in PCA (**Section 1.1.1**) we see the mappings here are identical.

Due to the equivalence with PCA, the PCs and eigenvalues may be extracted from $\mathbf{W}_{\mathrm{PCA}}$ using the SVD. The left singular vectors correspond to $\mathbf{Q}_k$, and the singular values correspond to $\boldsymbol{\Lambda}^{\frac{1}{2}}$. $\mathbf{R}$ may also be separated out since it corresponds to the right singular vectors; this may be of use if we have performed inference by maximising the likelihood numerically, since $\mathbf{R}$ would then be arbitrary. $\mathbf{R}$ can always be separated from $\mathbf{W}_{\mathrm{PCA}}$, $\mathbf{W}_{\mathrm{ML}}$ or $\mathbf{W}_{\mathrm{MAP}}$ in this way, so we consider only the case of $\mathbf{R} = I$ without loss of generality.

3.5 Conclusion

StPCA is a linear latent variable model with a Gaussian Process prior placed on the loadings, which implicitly places a prior over the covariance structure of the model. One often has knowledge about how a set of features relate to each other; if the features are pixels in an image or points in a time-series, we may expect the features which are closer in space/time to be more highly correlated. Such prior knowledge is not typically incorporated into traditional feature learning techniques such as PCA, which assume a-priori that features are independent. Taking advantage of this prior knowledge may produce a more accurate model, especially if the data are small-sample and high dimensional.

The StPCA prior is specified via choice of covariance function. This is a convenient method of prior specification since covariance functions are often interpretable, with interpretable hyper-parameters. Inference is performed by finding the MAP parameters, and then constructing an approximate Gaussian posterior centred at the MAP. Finding the MAP is performed using generalised EM, where the maximisation step increases the expected complete log posterior, but does not fully maximise it. This partial maximisation step involves setting up a Sylvester equation which is always guaranteed to give a solution.

After finding the MAP parameters, the Laplace approximation is used to

produce a Gaussian approximation to the posterior. An analytic form for the Hessian at the MAP is used to produce the covariance matrix for the approximating Gaussian, and the diagonal of this gives the approximate posterior variance for each parameter.

In performing the Laplace approximation, we also obtain an approximate marginal likelihood. Model selection may be performed by maximising this, which can aid selection of the latent dimensionality, covariance function and covariance function hyper-parameters. Bayes factors are well calibrated, so the amount of support for one model over another may be quantified.

StPCA is related to the well-known and understood models PCA and PPCA. Considering the limit of an infinitely broad StPCA prior, the PPCA model is exactly recovered. Consequently, PCA may also be recovered in the limit of the 'noise' parameter σ^2 going to zero.

Chapter 4

Structured PCA: Results

4.1 Synthetic Data

To test StPCA, synthetic data are simulated from the StPCA model, and StPCA is fitted back to the data. It is expected that StPCA would outperform any other technique in recovering an accurate model from the data. This exercise is not to judge the effectiveness of StPCA over other techniques, but to test its performance in a best-case scenario, to test how misspecification of $\mathbf{K}_\beta$ impacts performance, and to test that the implementation works as expected. The ability to recover hyper-parameters is of particular interest, since approximations have been made use of in hyper-parameter inference, making the model approximate.

Data Generation

Each sample of the synthetic data has the structure of a 21×21 image ($d = 441$). A single dataset contains $n = 15$ such images constructed by using a fixed $\mathbf{W}$ drawn from the prior where $k = 4$, and an i.i.d latent variable drawn from a multivariate unit Normal. 50 such datasets are generated and test performance is computed across these. The reason for this is that a single dataset has a fixed $\mathbf{W}$, but we would like to see performance across many instances of $\mathbf{W}$.

Each feature in the data represents a pixel value, where pixel i has coordinates $\mathbf{t}_i \in [0, 1]^2$. The covariance function used in generating the data is the Noisy Squared Exponential (NSE, **Section 1.3.2**) with hyper-parameters $\sigma_k^2 = 2$, $\ell = 0.2$, $\sigma_n^2 = 10^{-6}$. For the noise parameter we use $\sigma^2 = 1.8$, and for simplicity we use a mean μ of zero. Examples of this synthetic data are given in **Figure 4.1**; each row of images is the first 5 samples from within a single dataset. The dataset-specific structure can be seen as common directions of variance with each row.

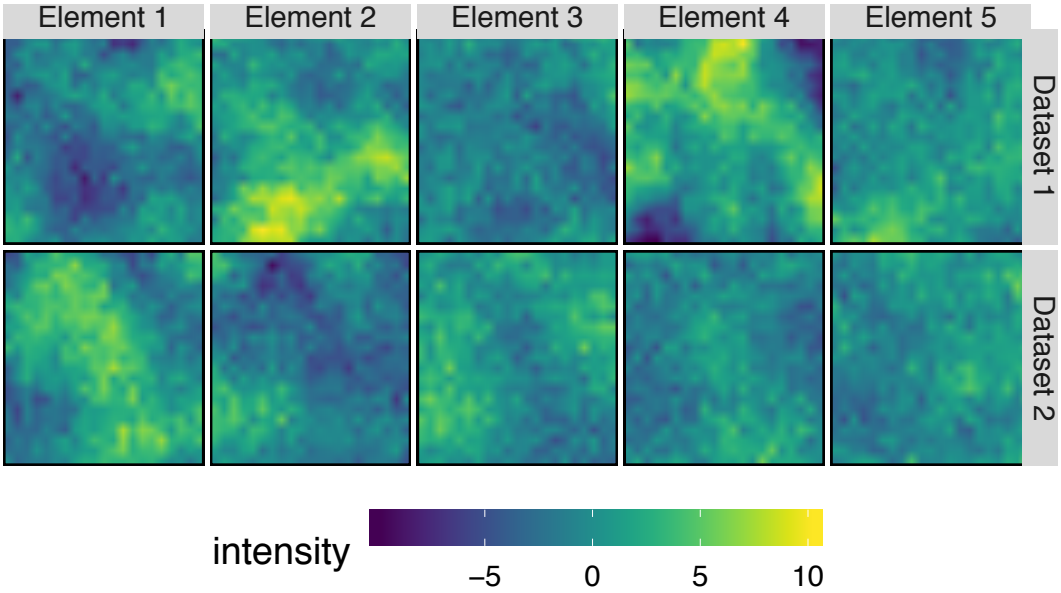


Figure 4.1: *The first 5 elements of the first (top) and second (bottom) datasets generated. Common patterns can be seen within each dataset, which are a result of there being only 4 smooth latent patterns of which each sample is a linear combination plus noise.*

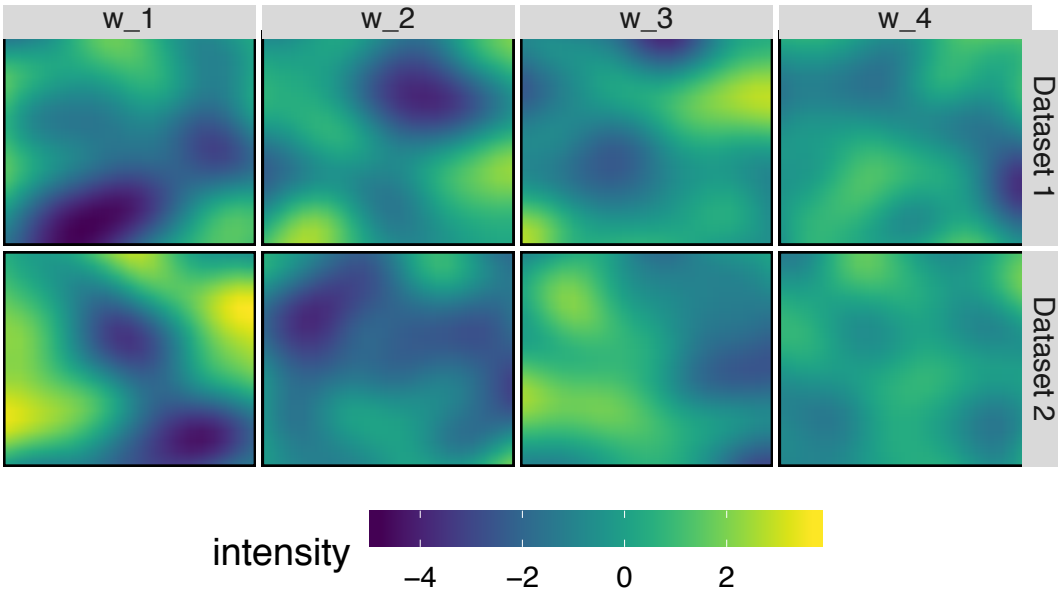


Figure 4.2: *Each column of the loadings matrices used to generate the synthetic data in* ***Figure 4.2*** *may also be displayed as an image. Each image in* ***Figure 4.2*** *is made up of a linear combination of the loadings displayed here, plus white noise.*

The data are simulated from StPCA using ancestral sampling. β, k, σ^2 are fixed at the values given above. $\mathbf{W}$ is then drawn from $p(\mathbf{W}|\beta)$ and $\mathbf{V}$ is drawn from $p(\mathbf{V}) = \mathcal{N}(0, I)$. The samples are then simulated from the likelihood $p(\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2) = \mathcal{N}(\mathbf{W}\mathbf{v}_i, \sigma^2 I)$. The procedure for generating the data is given algorithmically in **Algorithm 4.1**.

For the simulated data we would ideally like to use the SE covariance function to produce perfectly smooth loadings, and thus the samples would be smooth plus noise σ^2. However, simulating from $p(\mathbf{W}|\beta)$ requires inverting $\mathbf{K}_\beta$, which, in the case of the SE, is semi-definite and thus non-invertible. Using the 'noisy' SE (NSE) adds σ_n^2 to the diagonal, making $\mathbf{K}_\beta$ positive-definite and thus inversion possible. We choose the small value $\sigma_n^2 = 10^{-6}$ so we can simulate $\mathbf{W}$, but the columns are only contaminated with a low level of noise.

Covariance Structure Estimation

Here we test the ability of StPCA to uncover the subspace the synthetic data lie on. StPCA is compared to PCA and PPCA in this task, and StPCA is fitted with both the Rational Quadratic (RQ) and Tapered Squared Exponential (TSE) covariance functions. The data are generated using the SE, which is a special case of the RQ (when $\alpha \to \infty$). The taper length is set to 0.2 where each sample is a 1×1 box, so the TSE cannot learn parameters reducing it to the SE, but $\mathbf{K}_\beta$ is sparse.

The 4 techniques (StPCA (RQ), StPCA (TSE), PCA, PPCA) are all linear latent variable models, so produce a matrix of loadings defining the modelled subspace the data lie on. For a single dataset, each of the 4 techniques is fit with $k = 4$. We then compute the largest principal angle (**Appendix A.1**) between the

Algorithm 4.1: The process for generating the data

$\sigma^2 \leftarrow 1.8$;
$k \leftarrow 4$;
$n \leftarrow 15$;
$\beta \leftarrow \{\sigma_k^2 = 2, \ell = 0.2, \sigma_n^2 = 10^{-6}\}$;
$(\mathbf{K}_\beta)_{ij} \leftarrow k_{\text{NSE}}(||\mathbf{t}_i - \mathbf{t}_j||; \beta)\ \forall\, i, j \in 1 \cdots d$;
foreach $j \in \{1 \cdots 50\}$ **do**
  Draw $\mathbf{w}_i^j \sim \mathcal{N}(\mathbf{0}, \mathbf{K}_\beta)\ \forall\, i \in 1 \cdots k$;
  Draw $\mathbf{v}_i^j \sim \mathcal{N}(\mathbf{0}, I)\ \forall\, i \in 1 \cdots n$;
  Draw $\mathbf{x}_i^j \sim \mathcal{N}(\mathbf{W}\mathbf{v}_i, 1.8I)\ \forall\, i \in 1 \cdots n$;
end
return $(\mathbf{W}^j, \mathbf{V}^j, \mathbf{X}^j)\ \forall\, j \in 1 \cdots 50$;

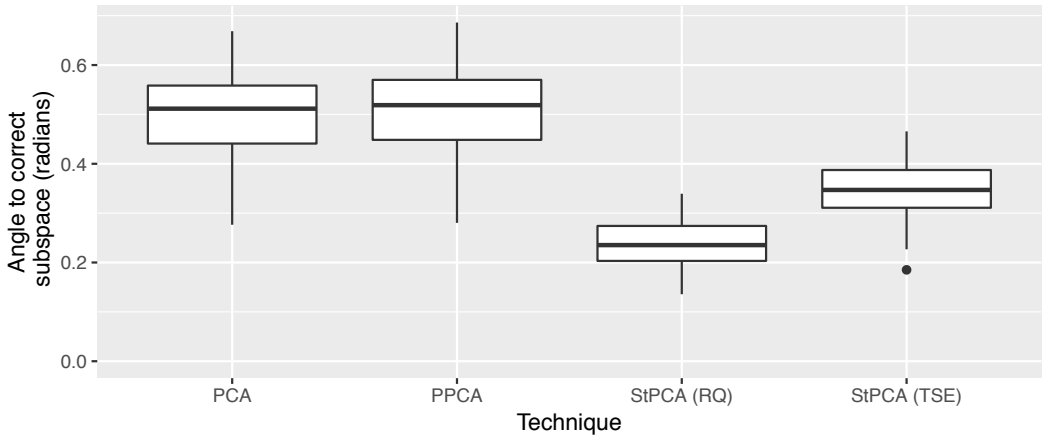


Figure 4.3: *Distribution of the largest Principal Angle between learned principal subspace and real principal subspace over 50 synthetic datasets. Lower angles indicate that the learned subspace is closer to the true subspace. StPCA with either covariance function outperforms PCA and PPCA. The RQ more flexible RQ covariance function outperforms the less flexible TSE.*

estimated subspace and the true subspace, which measures how different the subspaces are. Repeating this over all 50 synthetic datasets shows us the distribution of the largest principal angle for each technique, which is shown in **Figure 4.3**. It can be seen that StPCA with either covariance function outperforms both PCA and PPCA at subspace recovery, as the StPCA angles are lower. The RQ covariance function shows better performance than the TSE, which is due to the TSE being insufficiently flexible. In all cases the RQ learns hyper-parameters such that it closely approximates the SE.

The PCs/StCs may also be compared visually, as in **Figure 4.4**. Here is shown the first StC and first PC recovered from the first two synthetic datasets. These are compared to the ground truth direction of greatest variation of the data generating distribution. One can see that the ground truth and the two StCs are smoother than the PCs learned by PCA/PPCA.

Structured Component uncertainties

In **Figure 4.4**, there are regions which have been recovered well and regions which deviate substantially from the ground truth. Since StPCA returns a full (approximate) posterior, we can consider the certainty with which each pixel has been learned.

Figure 4.5a shows the absolute difference between the generating $\mathbf{w}_1$ in the first synthetic dataset, and the $\mathbf{w}_1$ as recovered by StPCA fitted to this dataset. **Figure 4.5b** shows, for each pixel, the standard deviation of the posterior over that

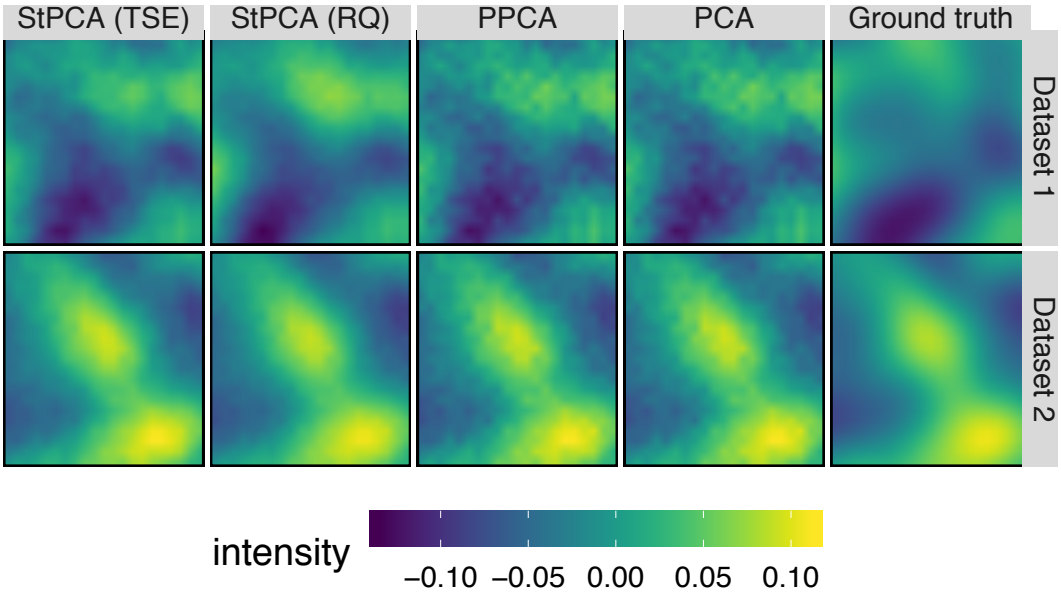


Figure 4.4: *The first column of* $\mathbf{W}$ *(i.e., the direction of most variation) as learned by StPCA, PPCA and PCA. The right-hand column shows* $\mathbf{w}_1$ *used in the underlying generative process. Greater similarity between the learned* $\mathbf{w}_1$ *and the true* $\mathbf{w}_1$ *indicated better performance. We can see that the learned* $\mathbf{w}_1$ *for PPCA and PCA is rougher than that learned by StPCA and the ground truth.*

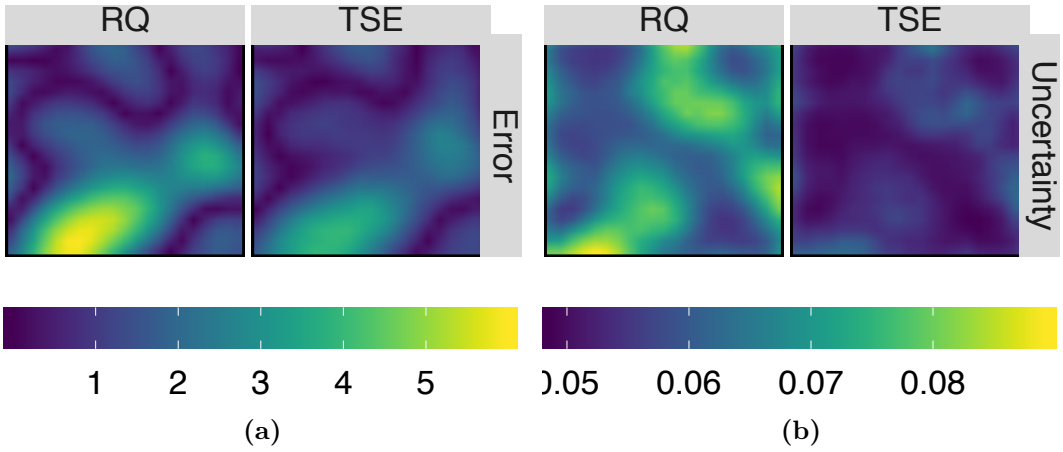
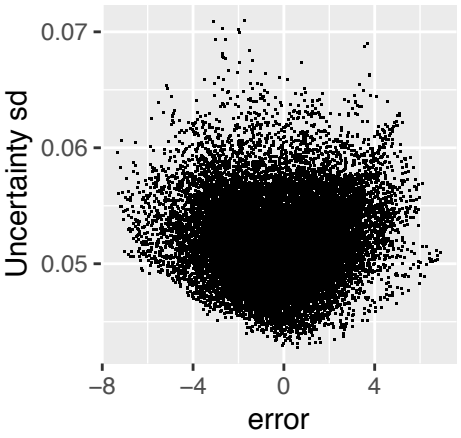
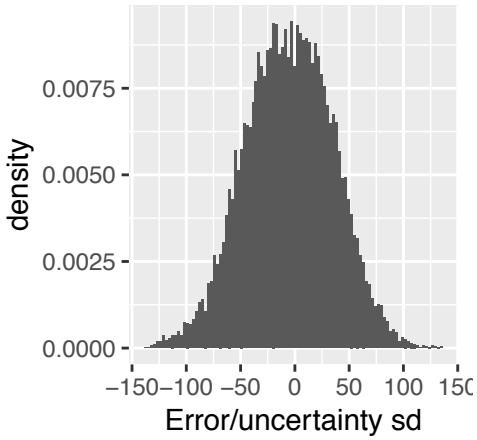


Figure 4.5: ***(a)****: Error in the first StC, as measured by the distance between each element and the corresponding element of the ground truth PC. This is shown for both the RQ and TSE covariance functions.* ***(b)****: 1 standard deviation of the uncertainty at each element of the StC. Uncertainty is underestimated, but correlates with error. The RQ and TSE covariance functions produce similar loadings, but differ in posterior uncertainty, with larger uncertainty assigned to the RQ.*



(a) *Larger posterior variance correctly predicts a larger expected error. Each plotted point corresponds to a single element of the inferred* $\mathbf{w}_1$ *over all 50 datasets. The* x*-axis shows the difference between the inferred and true value, and the* y*-axis shows the posterior standard deviation around the inferred value.*



(b) *Uncertainty is underestimated in StPCA. Dividing the error (inferred values of* $\mathbf{w}_i$ *minus ground truth) by the posterior variance would produce a standard Normal if the posterior variance were correct. The variance of this distribution is greater than 1, showing uncertainty is underestimated.*

Figure 4.6: *Posterior uncertainty is overestimated, but associated with high error.*

pixel. One can see that the regions which are learned accurately are associated with lower posterior uncertainty, and vice-versa.

Comparing the error to the posterior uncertainty, it can be seen that the regions of high error occur at regions of high uncertainty. This illustrates StPCA being able to correctly identify which elements of the StCs are accurately learnt, and which may deviate from the ground truth.

Larger reported uncertainty correctly predicts larger expected discrepancies between the estimated and true values of $\mathbf{w}_1$ (**Figure 4.6a**). However, the uncertainties reported are underestimates (**Figure 4.6b**). Care must thus be taken when interpreting the posterior variance; although one can see the relative confidence with which elements of $\mathbf{W}$ have been learned, the absolute confidence on each element may be misleading.

The Empirical Bayes approximation will contribute to the uncertainty underestimate, since the approximation states that we are extremely certain of the hyper-parameter value, placing all posterior mass on a single point $\hat{\beta}$. An exact posterior would take the uncertainty over this hyper-parameter estimate in to account. The contribution of the Laplace approximation to the uncertainty underestimate is unclear. However, our further approximation of the posterior Hessian being block-

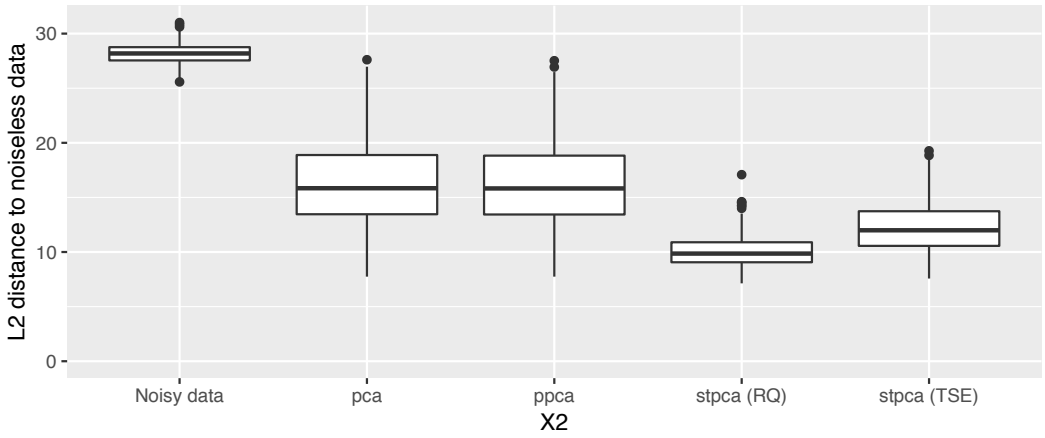


Figure 4.7: *Distribution of reconstruction errors between noiseless data and data which has been de-noised by each of the techniques. All techniques do better than no de-noising (left hand box). PCA and PPCA perform similarly, which is expected as they learn the same principal subspace. StPCA outperforms PCA/PPCA with both the Squared Exponential or Rational Quadratic covariance function, and the RQ performs the best.*

diagonal ($\mathbf{H}_\beta \approx \tilde{\mathbf{H}}_\beta$) does not change the uncertainties in **Figure 4.5**, since we only plot the variances on each pixel, corresponding to the diagonal of $\mathbf{H}_\beta$, and the off-diagonal elements are not plotted.

De-noising

PCA is often used in de-noising; the first k PCs are used to approximately capture the signal, and the remaining $d - k$ are discarded. Transforming the data to k-dimensional latent representations then back to d-dimensional data space gives a de-noised version of the data. StPCA can also be used to de-noise in this manner, and may outperform PCA if the prior is appropriate.

Using the previously generated synthetic data, we de-noise the data with PCA, PPCA and StPCA using $k = 4$. We then calculate Euclidean distances between the de-noised data and the noiseless synthetic data. This produces a distribution of reconstruction distances for each de-noising technique, which are shown in **Figure 4.7**.

The de-noised data are closer to the original signal for all de-noising techniques. PPCA and PCA perform similarly, whilst StPCA outperforms both. Again, the RQ covariance function produces a better result than the TSE, indicating it has fit the data better and the taper is too restrictive.

Figure 4.8 shows a single sample being de-noised. The StPCA de-noised samples are visually (and by ℓ_2 distance) more similar to the original sample than

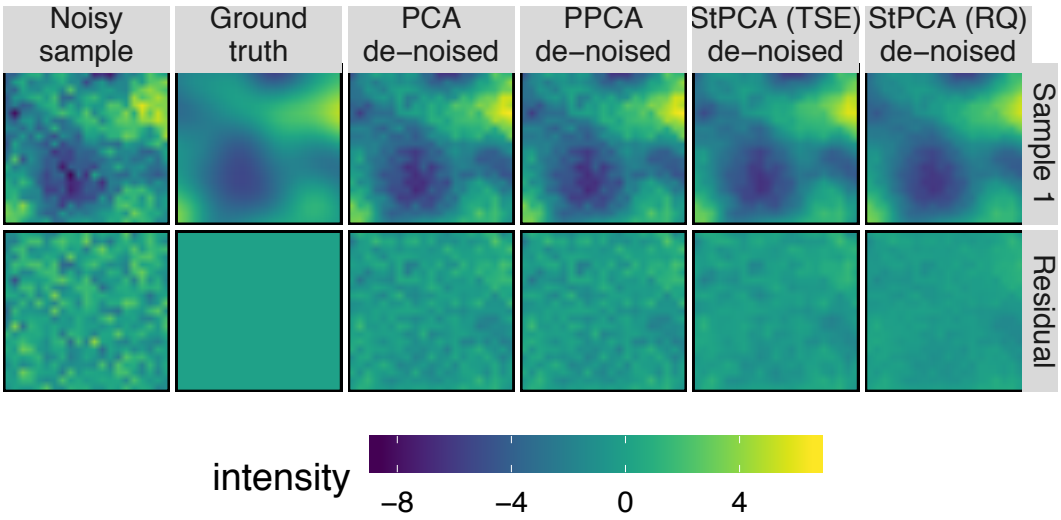


Figure 4.8: *Top row, left to right: the noisy and noiseless version of the first sample from the first dataset generated, along with a version de-noised by each algorithm. Bottom row: the images from the top row with the noiseless sample (row 1, column 2) subtracted, for the purpose of showing the error. The norm of the error for PCA and SPCA is larger than for StPCA.*

the PCA/PPCA de-noised samples.

Discrete Model Selection

We compare log Bayes Factors for selecting the latent dimensionality k and covariance function. StPCA is fitted to the synthetic data with values of k from 1 to 10, and log Bayes Factors are computed. This is done for the TSE and RQ functions, and we also include the (ground truth) SE for comparison. We consider Bayes Factors instead of marginal likelihoods so we are able to quantify the amount of evidence for one model over another. The Bayes Factors are normalised to the maximum marginal likelihood model, so the highest scoring model obtains a log Bayes Factor of zero.

The log Bayes Factors are shown in **Figure 4.9**. It can be seen that with all covariance functions the true dimensionality of 4 is identified. Although the SE and RQ covariance functions appear to perform similarly, the Bayes' Factor for the RQ covariance function at $k = 4$ is 8×10^{-29}, which is very strong evidence for the true SE covariance function.

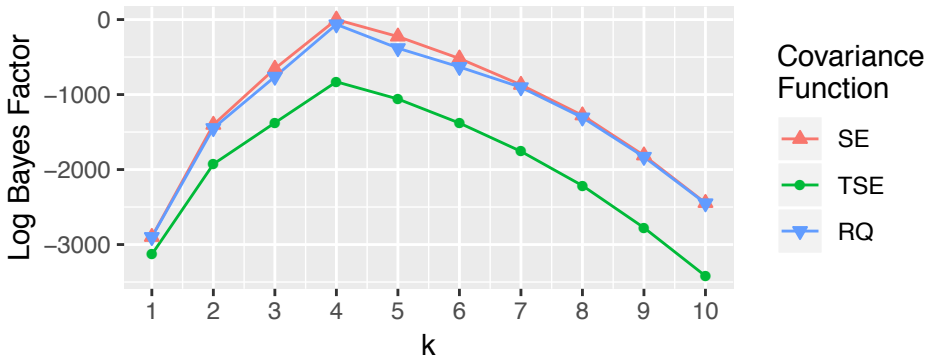


Figure 4.9: *Log Bayes Factors (normalised to the maximum marginal likelihood model) against k for StPCA with the 3 covariance function: Squared Exponential (SE), Tapered Squared Exponential (TSE) and Rational Quadratic (RQ). All models select the true value of k, and the true SE covariance function is selected. Visually the RQ performs similarly, but the Bayes Factor for the RQ with $k = 4$ is 8×10^{-29}, which is very strong evidence for the SE.*

Hyper-parameter tuning

To test hyper-parameter recovery, we fit the StPCA model with SE covariance function back to the generated data and compare the learned hyper-parameters against the known true ones. If StPCA inference were exact, the mean of the hyper-parameters inferred over many such datasets should be equal to the true set of generating hyper-parameters. However, the inference is approximate due to the use of the Laplace approximation. It is thus of interest to see how this impacts hyper-parameter learning. **Figure 4.10** compares the learned and ground truth hyper-parameter values. Also shown are the initial values used, which were obtained using the parameter initialisation described in **Section 3.3.2**.

Looking at the diagonal of **Figure 4.10**, it can be seen that the learned length scales ℓ and noise variances σ_n^2 are centred around the generating value. The learned signal variances σ_f^2 do not centre around the generating value, showing that in every case we learn a signal variance that is too low. This shows that we do not recover the data-generating hyper-parameters in expectation, so should be wary about drawing conclusions from hyper-parameter inferences. However, the initialiser for the signal variance appears to be good for these particular datasets. The other two initialisers give starting values further from the true values, but still in a sensibly close range such that the hyper-parameter optimisation finds a good mode.

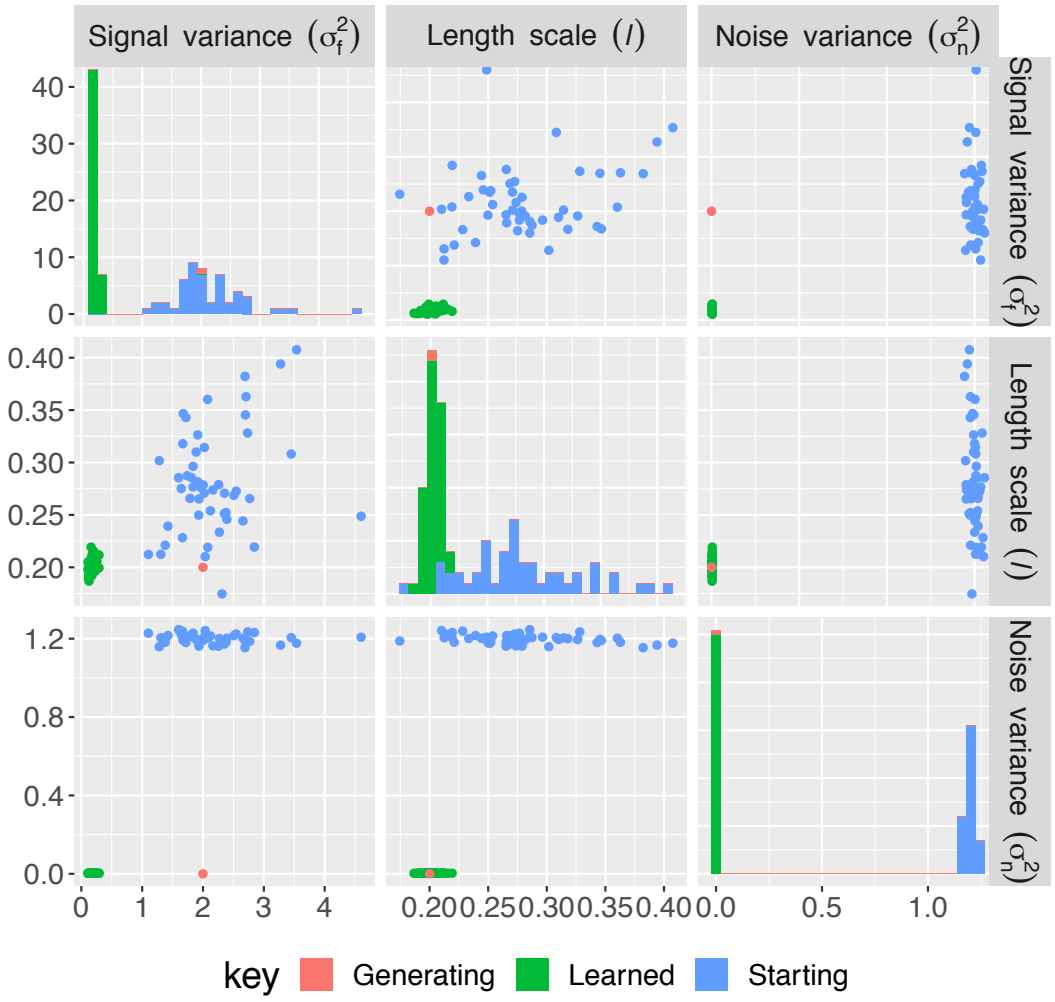


Figure 4.10: *Hyper-parameters: ground truth, initialising and learned values. The learned length scale is close to the true value, but the learned signal variance is an underestimate and the noise variance is overestimated.*

4.2 FAIMS Data

Here we investigate the performance of StPCA on real data. We use the IBD dataset from **Section 2.3.2**, which contains FAIMS measurements of the breath of 53 IBD patients and 11 healthy volunteers.

The full FAIMS data are too high dimensional for StPCA to easily process due to the $\mathcal{O}(d^3)$ scaling associated with inverting the matrix $\mathbf{K}_\beta$. For this reason we restrict the analysis to a subset of the pixels and compare to other techniques. However, the raw FAIMS data are over-sampled, so some level of feature selection is justified.

We compare 4 feature learning models – StPCA, PPCA, ICA and SPCA. PPCA, ICA and SPCA are all linear latent variable models and have been described in **Section 1.1.1**. Each model learns a mapping from high dimensional feature space to a low dimensional latent space, and any observed sample may be translated to a latent representation. We evaluate the quality of the latent spaces produced by the predictive accuracy obtained when using the latent representations as features in a classification task. The task chosen is to distinguish between IBD-positive and IBD-negative samples, and the metric used to evaluate this is 10-fold cross-validated AUC. AUCs are computed for a number of sparsity levels of the input; i.e., we retain different numbers of input pixels and compare how the 4 techniques perform at these different sparsity levels.

4.2.1 Pipeline

The results presented are from a single run of a pipeline described here. The data are first pre-processed, removing a large number of irrelevant features. The pre-processing stage is discussed in more detail below.

We then perform an analysis which performs model selection and fitting for each of the 4 models, then measures predictive capability using 10-fold cross-validated AUC. This analysis is performed 8 times, each using a different sized subset of the original features, with subset sizes equally spaced between 20 and 330. The purpose of this is to investigate how StPCA compares to other techniques as the input becomes more sparse.

To keep variance between runs to a minimum, once a feature has been removed at a given sparsity level, it remains removed for all higher sparsity levels. This removes the possibility of a high sparsity level retaining more informative pixels than a lower sparsity level and subsequently producing a higher performing classifier. This is achieved by removing a subset of the remaining pixels each time

Algorithm 4.2: Pseudo-code for pipeline to evaluate StPCA performance against PPCA, ICA and SPCA.

Input: Data $\mathbf{X}$, labels $\mathbf{y}$, feature learning models `models`

Pre-process $\mathbf{X}$; retain 330 features
foreach *sparsity in sparsityLevels* **do**
  foreach *model in models* **do**
    $\mathbf{X}_\text{s} \leftarrow$ drop features from $\mathbf{X}$ to correct `sparsity` level
    $\mathbf{V} \leftarrow$ feature transform $\mathbf{X}_\text{s}$ using `model` with hyper-parameter tuning
    for *fold in* $1 \cdots 10$ **do**
      $(\mathbf{V}_\text{tr}, \mathbf{y}_\text{tr}), (\mathbf{V}_\text{te}, \mathbf{y}_\text{te}) \leftarrow$ split $\mathbf{V}, \mathbf{y}$ into training/test set
      Train Gaussian process classifier on $\mathbf{V}_\text{tr}, \mathbf{y}_\text{tr}$
      $\mathbf{y}^*_\text{te} \leftarrow$ predict labels for $\mathbf{V}_\text{te}$
      Compute auc$(\mathbf{y}_\text{te}, \mathbf{y}^*_\text{te})$
    end
  end
end
return AUCs for each `sparsity`, `model`, `fold`

the sparsity level is increased.

The feature transformation and model selection for k (and also the sparsity parameter λ for SPCA) is performed prior to splitting the data into the training and test set in cross-validation. Although this is using data from the test set in constructing the feature space, this is statistically valid since it does not make use of the labels. Such a procedure could still be applied in a real world medical setting: upon collecting a new sample, a feature learning stage could include this sample without needing to know the diagnostic outcome. An alternate approach would be to learn the feature transformation on the training set and apply this to the test set. Our approach was chosen to ease the computational burden, as only 1 model selection stage must be performed instead of 1 per fold.

Following the feature transformation, the AUC is computed using 10-fold cross-validation. Pseudo-code for this procedure is presented in **Algorithm 4.2**. Predictions are made using the Random Forest (RF) and Gaussian Process Classifier (GPC).

Data pre-processing

To remove irrelevant features, pixels with a standard deviation below 0.01 are removed. This procedure has already been established to work well on previous datasets, and is used by an existing FAIMS data processing pipeline discussed in **Section 2.2.2**. Observing the standard deviation of each pixel, there is a high frequency

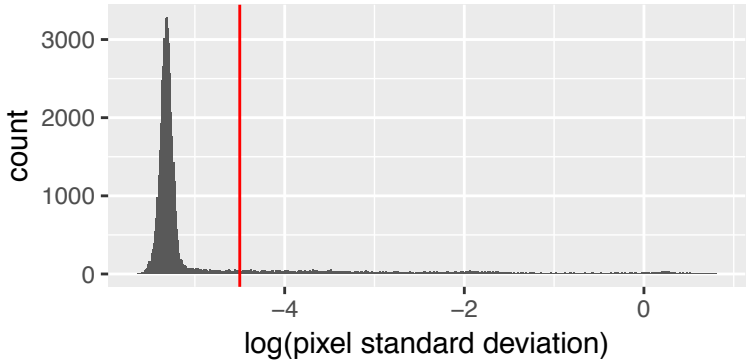


Figure 4.11: *Histogram of the log standard deviation of pixel intensity values. The low-variance peak towards the left corresponds to pixels which only vary between samples due to measurement noise, and contain no signal relating to the actual sample. All pixels with a standard deviation below that indicated by the red line at 0.01 are dropped.*

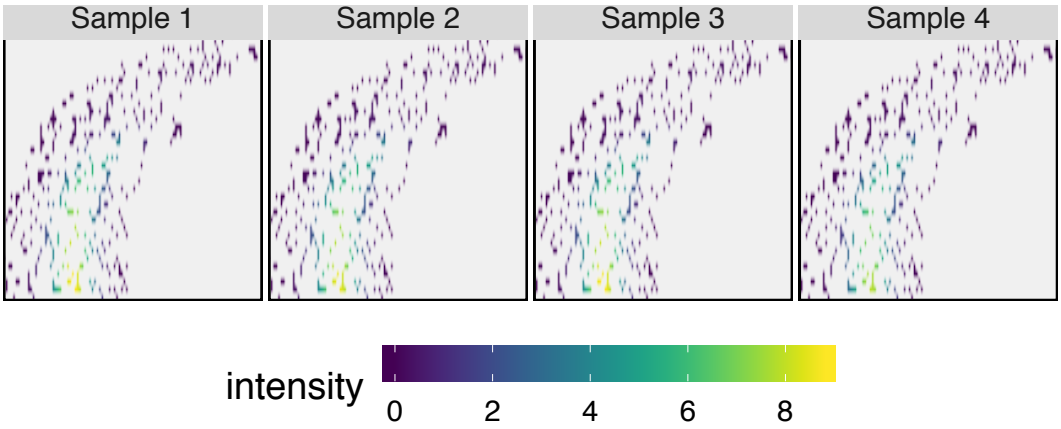


Figure 4.12: *FAIMS data after pre-processing. Low variance pixels are removed, then pixels with no neighbours and edge pixels are removed. A random subset of 330 of the remaining pixels are retained.*

of pixels with similar low-level variation which stems only from measurement noise, and not from intra-sample variation (**Figure 4.11**). The threshold was chosen to drop all pixels with a standard deviation near this measurement noise value. Following this, any pixels with no neighbours are also dropped, as are edge pixels. From the remaining pixels, a random subset of 330 are retained. Examples of samples post pre-processing are given in **Figure 4.12**.

Model Selection

For all models, model selection is performed for k, the latent dimensionality. The way this is performed is different per model, and is described below. SPCA requires

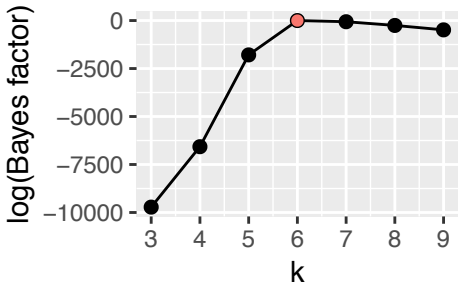


Figure 4.13: *StPCA is fit for values of k between 3 and 9, and Bayes factors computed with the highest-evidence model as the divisor. The highest evidence model ($k = 6$ in this case) thus has a log Bayes factor of zero and is highlighted here in red.*

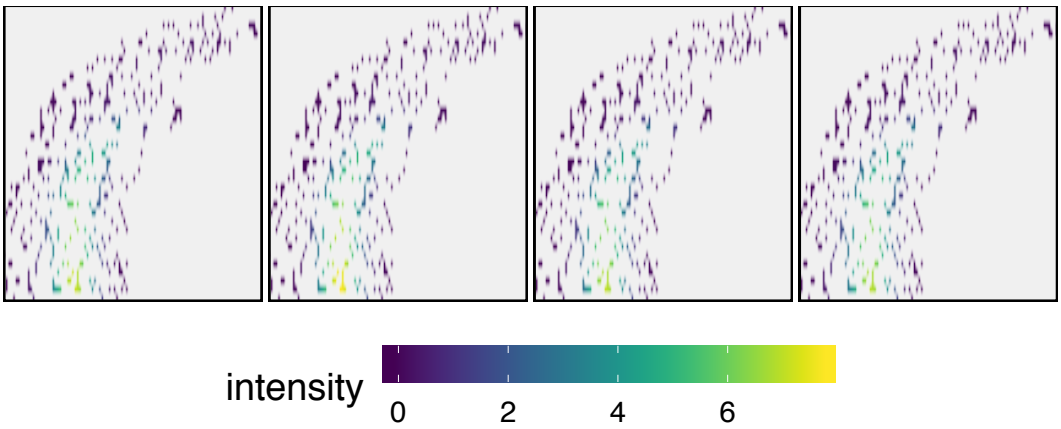


Figure 4.14: *Synthetic data simulated from the fitted StPCA model with $k = 6$ does not show anything pathological about the fitted model. The data show the characteristic high intensity region in the bottom centre, and variations are within sensible bounds.*

selection of the additional parameter λ controlling the sparsity of the loadings, which is selected with k in a 2d grid search.

Model selection for StPCA is performed by maximising the approximate marginal likelihood. A value of k between 3 and 9 is selected. StPCA is fit for each value of k with the noisy squared exponential covariance function, and β is tuned. Bayes factors are then computed between each model, normalised to the maximum marginal likelihood model.

As an example, we plot the log Bayes factors for each k in the case of StPCA being fit to the full 330 dimensional data in **Figure 4.13**. A value of $k = 6$ is selected. All Bayes factors other than that for $k = 6$ are very much smaller than 1, so a value of 6 is chosen with a high degree of certainty. To comment on the biological significance of this value would require deeper understanding of the

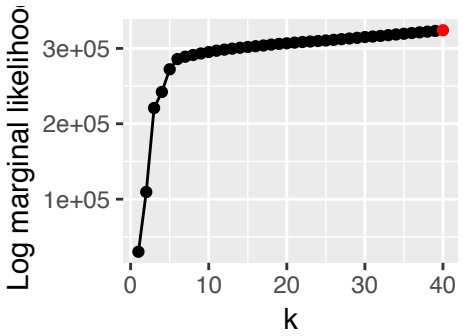


Figure 4.15: *Fitting PPCA to the data, the Laplace-approximated log marginal likelihood increases monotonically with k up until the largest value considered (40). This means that PPCA picks a very large value of k compared to other methods. Our prior knowledge about the FAIMS data tell us that it is unlikely that 40 independent directions of variation are supported by the data.*

compounds contributing to the variations captured by the model, as well as the biology underlying the production of these compounds, both of which are outside the scope of this thesis.

Since StPCA is probabilistic, we can simulate data from the fitted model and visually inspect it for deviations from how we expect a good model to behave. 4 draws from the fitted model are shown in **Figure 4.14**; these appear sensible, showing a high intensity region at the bottom centre as characteristic of FAIMS.

Model selection for PPCA is performed by fitting PPCA for for k in the range 1–40, and computing an approximation to the marginal likelihood for each fitted model. The model with the greatest marginal likelihood is selected.

The marginal likelihood approximation used is the Laplace approximation (**Section 1.2.1**), which is the same as used for StPCA. This is an important comparison to make as the model selection for PPCA and StPCA is the same except for the additional prior used in StPCA.

Again, we demonstrate this in **Figure 4.15**, plotting the log marginal likelihoods obtained for each k. In this case, the log marginal likelihood continues increasing as we increase k up to the maximum value, so $k = 40$ is selected.

Model selection in SPCA is performed by minimising an information criterion introduced by Hubert et al. [2016]. This is used to select both k as well as the SPCA sparsity parameter λ. The (k, λ) pair selected is that minimising the information criterion.

The information criterion is computed at a grid of (k, λ) values, with k

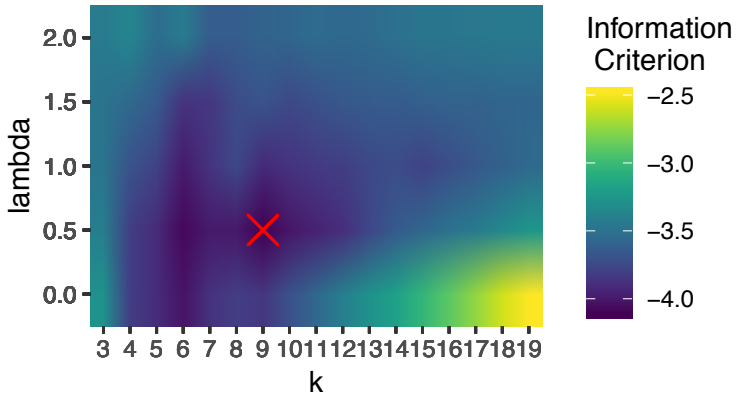


Figure 4.16: *The pair (k, λ) requires selection in SPCA, and this is done by minimising an information criterion [Hubert et al., 2016]. The information criterion is computed for a grid of (k, λ) values, and the pair minimising the information criterion is selected. Illustrated is the grid computed using the 330 dimensional data, where $k = 9$, $\lambda = 0.5$ have been selected.*

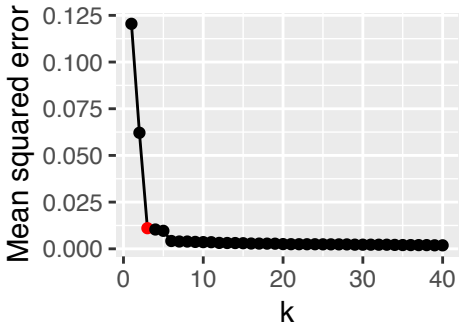


Figure 4.17: *Model selection with ICA. For each value of k a reconstruction of the data is computed and the mean squared error (MSE) between the real and reconstructed data calculated. The value of k is selected at which the MSE exhibits an 'elbow', defined as the greatest second derivative over the integers, which, in this case, is $k = 3$.*

between 3 and 19. 5 equally spaced values of λ considered within the range 0 – 2 are considered. A plot illustrating this is given in **Figure 4.16**, which was created by model-selecting SPCA on the 330 dimensional data.

Model selection for ICA is performed by computing cross-validated reconstruction error for values of k from 1 to 40, and selecting the value of k at the 'elbow'. Writing the mean squared error obtained at k as $\text{MSE}(k)$, we define the elbow as the point maximising

$$\text{MSE}(k-1) - 2\text{MSE}(k) + \text{MSE}(k+1) \tag{4.1}$$

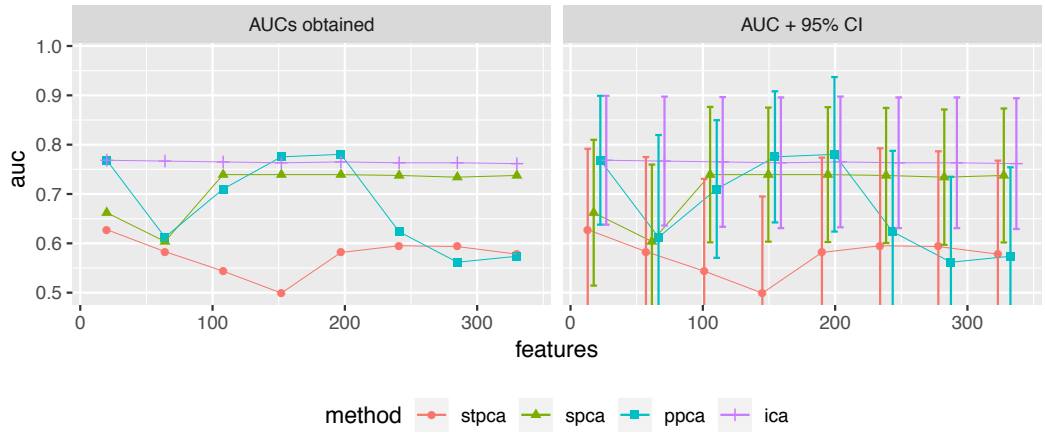


Figure 4.18: *AUCs obtained on FAIMS data with Gaussian Process Classifier in 10-fold cross-validation after dimension reduction with StPCA, SPCA, PPCA and ICA with model selection. The process is repeated with different sized feature subsets (x-axis). The two plots differ only in the inclusion of error bars. StPCA under-performs other techniques, but there is large uncertainty.*

which is the second derivative on the integers. As above, a plot illustrating this process is given in **Figure 4.17**, in which model selection is performed on the 330 dimensional FAIMS data and the selected value is $k = 3$.

Predictions

Predictions are produced using a Gaussian process classifier[1] This is done inside a 10-fold cross-validation. Transformation to feature space is done prior to the cross-validation so that feature learning only needs to be performed once.

4.2.2 Results

By running the pipeline described in the previous section and plotting the AUC against the number of input features, we obtain **Figure 4.18**. It can be seen that StPCA here under-performs SPCA, PPCA and ICA. However, the 95% confidence intervals are larger than the differences between feature extraction techniques, so this result is obtained with low certainty. The reason behind the large confidence intervals is that the data set is only small ($n = 32$).

Plotting the latent dimensionality selected by each technique at each feature count, we obtain **Figure 4.19**. Here we see StPCA and ICA are far more stable than SPCA and PPCA. PPCA uses the Laplace approximation to the marginal

[1]Implemented by the author as described in [Rasmussen and Williams, 2005]. R package hosted at `https://github.com/JimSkinner/gpclassifier`.

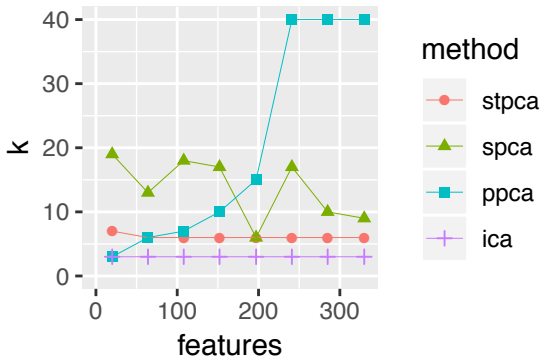


Figure 4.19: *Model selection is performed differently for each method, though always unsupervised. Both StPCA and PPCA maximise the Laplace approximation to the evidence, so it is interesting that PPCA selects larger k for larger d, but StPCA consistently picks $k = 6$, despite the similarity between models. ICA appears to compress the data well, having both the best performance in* ***Figure 4.18*** *and the lowest selected dimension.*

likelihood, as does StPCA. It is thus interesting that the selected values of k for PPCA diverge as the number of features increases, but this does not happen for StPCA. This could be due to the spatial prior in StPCA.

In the remainder of this section we probe these results with a few computational experiments. We replace the Gaussian Process Classifier used with a Random Forest to determine how much of the result is classifier-specific. We then disable model selection and fix k, allowing the model selection performance to be teased apart from the fitted model performance. Finally we re-run the entire pipeline with multiple seeds so to observe the variability in the result.

Replacing the Gaussian Process with a Random Forest classifier

Here we replace the Gaussian Process Classifier (GPC) with a Random Forest (RF) classifier. The reasoning behind this is that the RF is known to be a particularly robust classifier, and the poor StPCA performance may be due to the GPC not fitting well. This re-run uses the same seed as the original results, so the variation is not due to a different pixel subset in the data.

The results from using the RF are plotted in **Figure 4.20**. In this case all the StPCA AUCs become comparable to those from the other methods. As before, the 95% confidence intervals are still large compared to the differences between methods. The meaning of this result is that StPCA is not necessarily producing a worse feature representation for disease classification in this scenario, only that the particular classifier (GPC) failed to fit the features well.

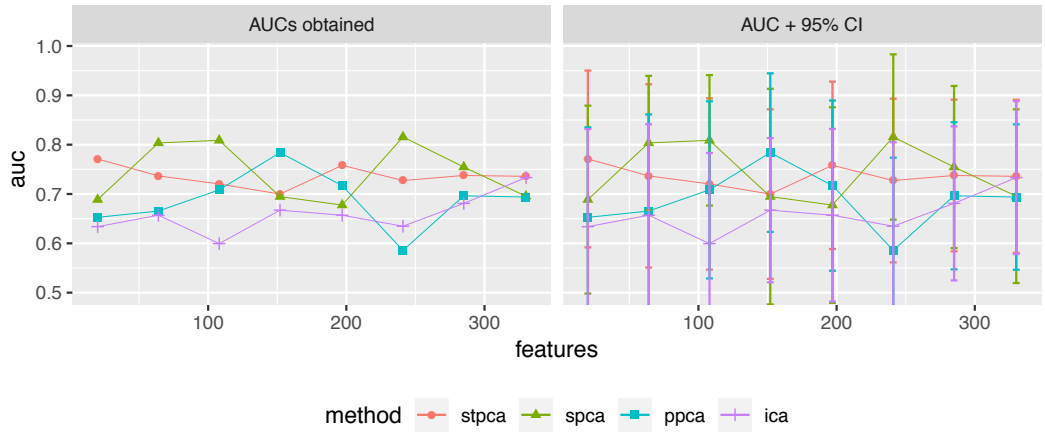


Figure 4.20: *By replacing the Gaussian Process Classifier (GPC) with the Random Forest (RF) for making the predictions used to compute AUCs, the performances between techniques become comparable. This suggests that the GPC may have had difficulty fitting the features produced by StPCA in* ***Figure 4.19****.*

Fixing k

For this experiment we disable model selection and fix k to 3 (the value selected by ICA) and 4. AUCs are again computed using the RF and GPC in 10-fold cross-validation, and are presented in **Figure 4.21**.

For the GPC, we see that StPCA performance improves for a latent dimensionality of 3 or 4. This is likely due to the GPC poorly fitting the higher dimensional model-selected feature space, so says little about the StPCA performance. For $k = 3$, we recover the original behaviour of ICA as expected. When $k = 4$, StPCA joins ICA/PPCA as a top-performing method. However, unlike these the performance does not drop for the highest feature count used, perhaps because of the additional structure imposed by the spatial prior.

For the RF, the performance of StPCA remains good for $k = 3$ and 4, whilst the PPCA and SPCA performance degrades. Both StPCA and ICA appear quite robust to the value of k, but StPCA outperforms ICA in every case.

From this experiment we can see that StPCA is able to effectively compress FAIMS data, producing accurate classifications even down to 3 dimensions. We also see that the GPC used is not always able to fit the StPCA feature space effectively. Comparing StPCA to PPCA is interesting to extract the influence of the structured prior, which we see gives a boost in performance in almost all cases when using the RF.

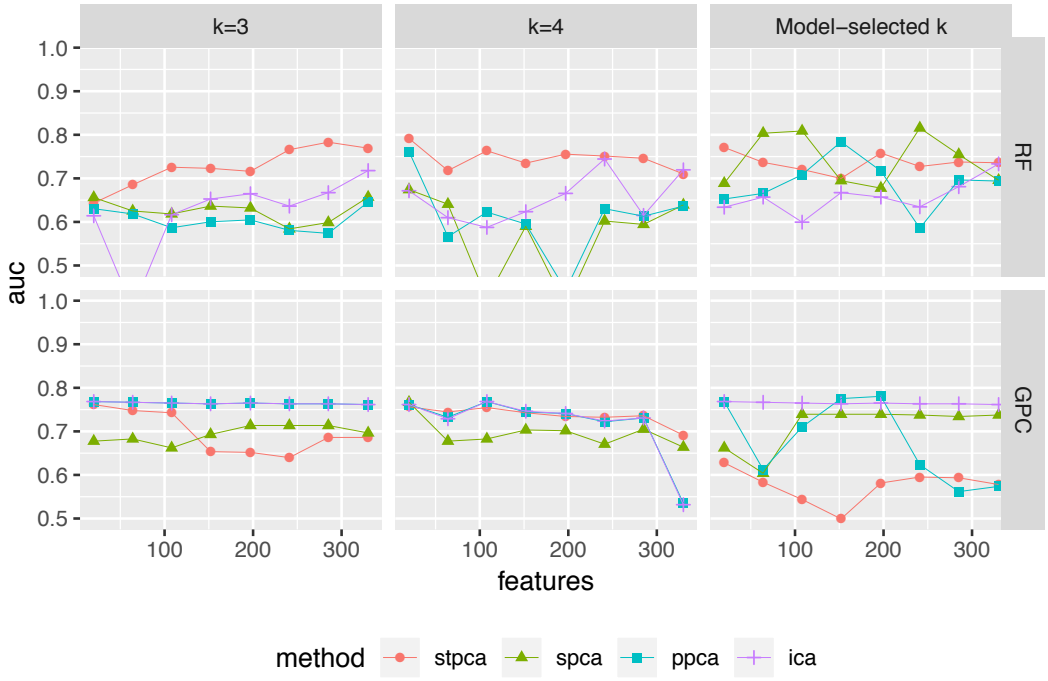


Figure 4.21: *We re-compute AUCs with k fixed to 3 and 4 (left two columns) with both the Random Forest (RF) and Gaussian Process Classifier (GPC). The poor GPC performance for on StPCA with model selection is likely due to the GPC failing to fit well, and we observe a better fit on a lower dimensional feature space. Looking at the RF, StPCA performs well with fixed $k = 3$ and 4, though not better than when model selection is enabled.*

Multiple restarts with different seeds

Finally we repeat the entire fitting and prediction procedure with multiple different seeds to observe the variability in the results. Since the model fitting is deterministic, the only stochastic part is the input feature subsetting. The only difference between runs is thus the subsets of features used.

In **Figure 4.22** we plot the AUCs obtained with the RF over multiple runs for each dimension reduction method. The mean AUC for StPCA is higher than all other techniques over the entire range of feature counts, and the StPCA AUCs are also more stable than all other techniques.

Looking at the selected values of k in **Figure 4.23**, we see StPCA and ICA are highly stable across runs and feature counts. The consistency in StPCA indicates a peaked posterior in k, since a very broad posterior would be reflected in an erratic value of k being selected. This is also the scenario in which the Laplace approximation is reliable. There is a small increase in selected dimensionality for StPCA for very low feature counts, which is surprising since for fewer features we

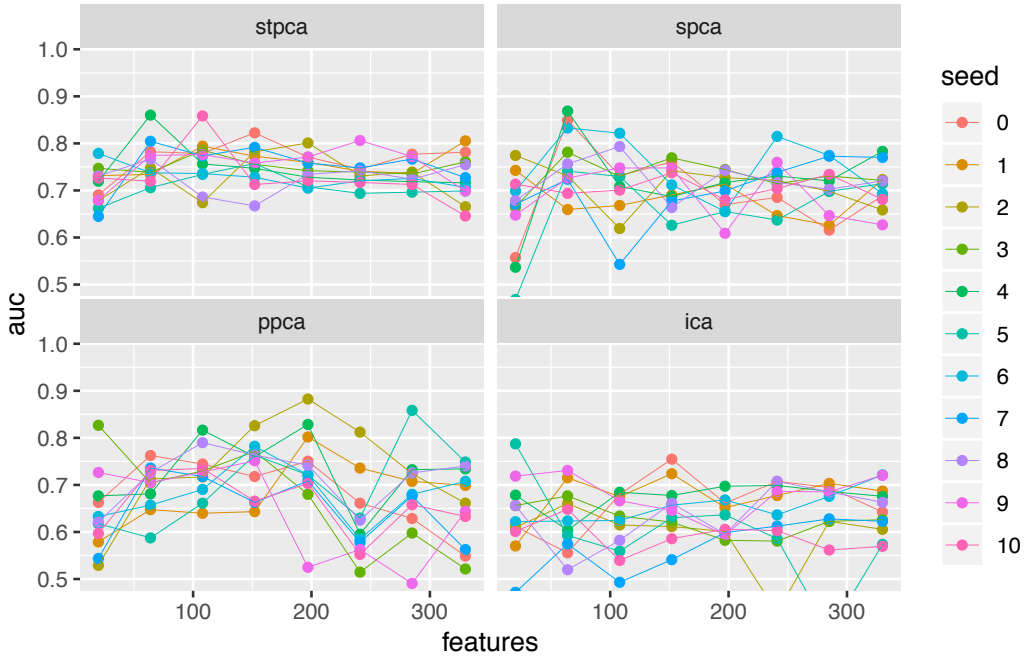


Figure 4.22: *Re-running with different seeds gives different RF AUCs due to feature subset variability. Compared to other methods, StPCA produces a greater mean AUC for all feature counts, as well as more stable AUCs.*

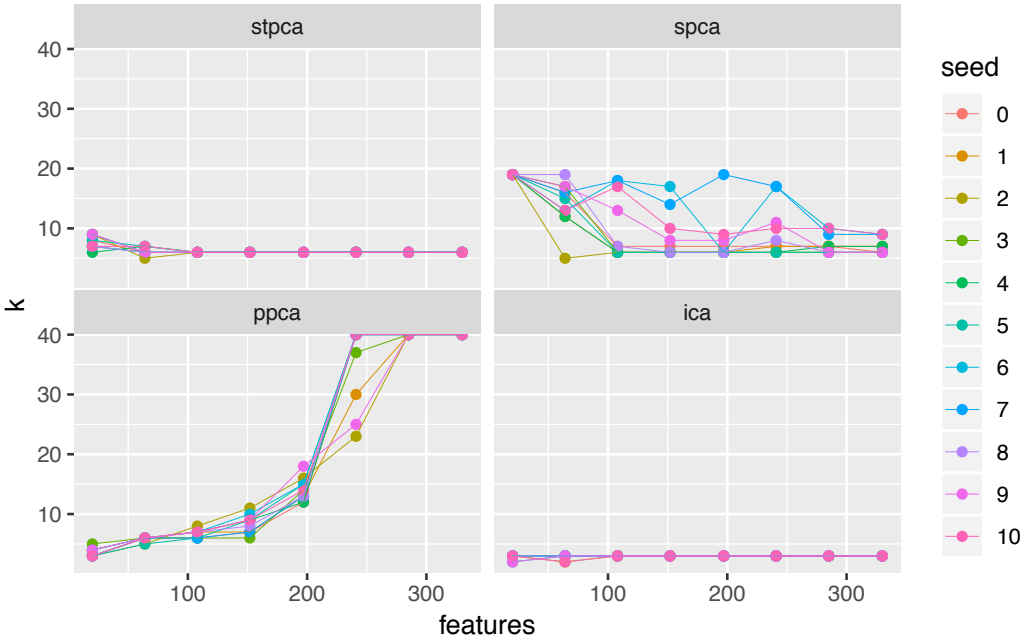


Figure 4.23: *The selected k for StPCA is stable, picking $k = 6$ over most of the feature count range. The increase in the selected k for low feature counts is notable, since once would expect a lower dimensionality to be selected. It is also interesting that the PPCA k increases with the feature count, but the StPCA k does not.*

would expect a simpler model to be selected.

PPCA reliably increases the size of the latent space with the feature count, but this increase model complexity does not appear to positively influence AUC. SPCA appears unstable in the selected latent dimensionality, despite the AUCs being stable, indicating a robustness to the selected k. ICA reliably produces the highest AUCs and lowest selected ks, so appears to be an effective dimension reduction technique for this data.

It is surprising that we do not see a consistent degradation in performance as the feature count is reduced. This indicates that only a small number of FAIMS pixels are sufficient for classification.

4.3 Conclusion

This chapter has been split into a section on fitting to synthetic data, and a section on fitting to FAIMS data. Synthetic data are drawn from an StPCA model, and StPCA, PCA and PCA are fit to the data. When simulating data from StPCA, the SE covariance function is used, but when fitting to the data we use the RQ and TSE covariance functions. We use a number of methods to evaluate the fitted StPCA, PPCA and PCA models.

The data generating process generates noiseless data on a k-dimensional subspace, then adds isotropic Gaussian noise to this. We compare methods by their ability to recover this subspace from data, and find that StPCA is better able to recover this subspace, despite having a misspecified covariance function. When plotting the direction of greatest variation for each model, we find that this is smoother for StPCA, reflecting the ground truth, than it is for PPCA and PCA.

StPCA has an advantage over PCA and PPCA of being able to produce an approximate posterior over the loadings instead of a point mass. We show that higher posterior certainty is correlated with lower variation from the ground truth, but uncertainty is underestimated in the particular case considered.

In de-noising a sample, StPCA also outperforms PPCA and PCA in terms of ℓ_2 distance between the de-noised sample and the ground truth noiseless sample.

StPCA produces an approximate marginal likelihood, which may be used for model selection. We illustrate that, on the data considered, maximisation of the approximate marginal likelihood in StPCA picks the ground truth covariance function, as well as the ground truth latent dimensionality.

We then apply StPCA to FAIMS data, reducing the data to a lower dimension with StPCA and producing predictions (IBD vs control) from this feature space.

Comparing StPCA to PPCA, SPCA and ICA, StPCA is found to produce the greatest AUCs on average when paired with the Random Forest classfier, and also displays the least AUC variance. Comparing StPCA to PPCA, this increase in performance must be due to the structured StPCA prior, since the is the only way in which StPCA and PPCA differ. We use both the RF and GPC classifiers in this section, but the GPC appears to not fit the StPCA feature space well.

The StPCA prior induced by the SE covariance function is quite crude. The SE encodes stationarity, but looking at FAIMS images shows clear non-stationarities, since there is always a plume around the same location and there is far more variation within this plume than around the image edges. It may be the case that a non-stationary covariance function can be constructed to specify this, possibly improving performance.

Chapter 5

Sparse Structured PCA

This chapter describes an extension to StPCA which produces a sparse loadings matrix. We name this extension Sparse StPCA (SpStPCA).

StPCA has been designed with the case of $d \gg n$ in mind, where there is typically high uncertainty and the addition of prior knowledge on the loadings may help learn a good model. We take this idea further with the introduction of sparsity. When dealing with $d \gg n$, assuming sparsity, i.e., that only a subset of the input features contain useful information, has proven to be useful [Hastie et al., 2015], leading to techniques such as Sparse PCA (SPCA). We thus investigate combining StPCA with sparsity, and see how this affects the ability to make accurate predictions on FAIMS data.

We begin with background in **Section 5.1**, starting with *subgradients* – a generalisation of the gradient often used in sparse modelling – and then discuss existing techniques in sparse modelling. The modelling assumptions and inference procedures for SpStPCA are covered in **Section 5.2**, and results of applying SpStPCA to FAIMS data are presented in **Section 5.3**.

5.1 Background

Sparsity is not a new concept; the Lasso [Tibshirani, 1996] is a well-known method of regularising a statistical estimator to produce sparse coefficients, which is desirable for a number of reasons. Presence of exact zeroes can reduce memory and computational requirements, and may improve interpretability. Lasso regularisation may also reduce the variance in the estimate of the coefficients at the cost of only a small increase in bias.

The Lasso computes a point estimate of the parameters which corresponds

to the posterior mode when the parameters have i.i.d Laplace priors. The Bayesian Lasso [Park and Casella, 2008] extends this by Gibbs sampling from the posterior, and also enables model selection of the sparsity hyper-parameter using Bayesian methods. The Bayesian Lasso was initially presented in a regression setting, but Guan and Dy [2009] use the Bayesian Lasso model to derive a Sparse Probabilistic PCA model using variational inference to report an approximate posterior.

A number of sparse latent linear models exist: SPCA uses sparsity in the loadings, whilst Dictionary Learning is sparse in the latent variables. It is very common to obtain sparsity via minimisation of an ℓ_1 norm, which has attractive analytic properties which are discussed below.

5.1.1 Subgradients and subdifferentials

In StPCA, finding the *maximum-a-posteriori* $\mathbf{W}$ is performed by setting the gradient of the negative expected complete log posterior to 0 and solving for $\mathbf{W}$. This works because the function is convex and differentiable, so any value of $\mathbf{W}$ with zero gradient is a global minimum. In this chapter we consider minimisation of convex but nondifferentiable functions, meaning that we cannot look for points with zero gradient since the gradient may not exist. To get around this we use the notion of *subgradients* and *subdifferentials*.

For a differentiable convex function, the tangent plane at any point always provides a lower bound for the entire function. At any nondifferentiable point the tangent plane is not defined. Instead we may consider the set of all planes (the "subtangent planes") which provide a lower bound.

In 1 dimension, the subdifferential of a convex function f at point x' is the set of all scalars z such that

$$f(x) \geq f(x') + z(x - x') \tag{5.1}$$

for all x in the domain of f. If z is an element of the subdifferential, we call it a subgradient. Generalising to p dimensions, a vector $\mathbf{z} \in \mathbb{R}^p$ is a subgradient of a convex function $f : \mathbb{R}^p \to \mathbb{R}$ iff

$$f(\mathbf{x}) \geq f(\mathbf{x}') + \mathbf{z}^\top(\mathbf{x} - \mathbf{x}') \tag{5.2}$$

Whenever f is differentiable at $\mathbf{x}'$, the subdifferential is equal to the singleton set containing the gradient. At nondifferentiable points, the subdifferential is a convex set of all subgradients.

An example nondifferentiable convex function which will be used throughout

this chapter is the absolute value function $f(x) = |x|$. The subdifferential in this case is

$$\frac{\partial f}{\partial x} = \begin{cases} \{+1\} & \text{if } x > 0 \\ [-1, +1] & \text{if } x = 0 \\ \{-1\} & \text{if } x < 0 \end{cases} \tag{5.3}$$

which can be written compactly as $\text{sign}(x)$. Here we use $\frac{\partial f}{\partial x}$ to indicate the partial subdifferential of f with respect to x, overloading the partial derivative operator. We continue this syntax throughout this chapter, and the intended operator should be clear from context.

Using subgradients, the condition for $\mathbf{x}'$ to be a minimum of a convex nondifferentiable function f is

$$\mathbf{0} \in \frac{\partial f}{\partial x}\Big|_{\mathbf{x}=\mathbf{x}'} \tag{5.4}$$

That is, the vector of all zeroes is a subgradient of f at $\mathbf{x}'$. In the case of $f(x) = |x|$, we can see from **Equation 5.3** that this is satisfied when $x = 0$.

5.1.2 The Lasso

The Lasso (Least Absolute Shrinkage and Selection Operator) is a method for estimating model parameters where the parameter estimates are sparse (contain exact zeroes). Here we will focus on the case of the Lasso being applied to a linear model, but the Lasso may be applied more generally. In the linear regression case, we have a matrix of predictors $\mathbf{V} \in \mathbb{R}^{n \times k}$ and vector of responses $\mathbf{x} \in \mathbb{R}^n$. Our aim is to fit the linear model $\mathbf{x} \approx \mathbf{V}\mathbf{w}$, where it is common to fit the vector of coefficients $\mathbf{w} \in \mathbb{R}^k$ using the least squares estimator

$$\underset{\mathbf{w} \in \mathbb{R}^p}{\arg\min} \ \frac{1}{2}||\mathbf{x} - \mathbf{V}\mathbf{w}||_2^2 \tag{5.5}$$

The least-squares Lasso estimator is obtained by considering the least squares estimator and subjecting this to the constraint that the parameter estimates lie inside the ℓ_1 ball of radius t:

$$\begin{aligned} &\underset{\mathbf{w} \in \mathbb{R}^p}{\arg\min} \ \frac{1}{2}||\mathbf{x} - \mathbf{V}\mathbf{w}||_2^2 \\ &\text{subject to } ||\mathbf{w}||_1 \leq t \end{aligned} \tag{5.6}$$

The solution to this problem often contains exact zeroes. $t > 0$ is a hyper-parameter, where smaller values produce sparser solutions.

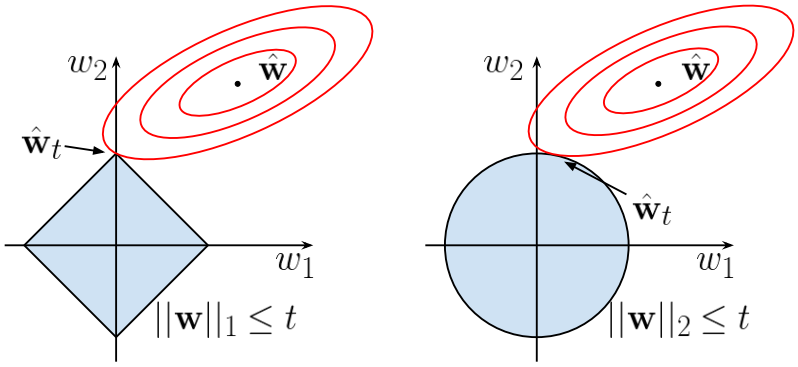


Figure 5.1: *Constrained ℓ_1 norm (left) and ℓ_2 norm (right). One can see that in the ℓ_1 constrained plot, the optimum value of $\boldsymbol{\beta}$ lies at the corner of the constrained region where $\boldsymbol{\beta}_2$ is equal to 0, and thus $\hat{\boldsymbol{\beta}}_t$ is sparse. This sparsity does not result from the ℓ_2 case. Figure adapted from [Hastie et al., 2015, Figure 2.2].*

Why is a sparse solution produced?

Figure 5.1 illustrates why the ℓ_1 constraint produces sparsity, but the ℓ_2 constraint does not. Unconstrained, the problem has the solution $\hat{\mathbf{w}}$, with red contours showing the squared error function to be minimised. When restricting $\mathbf{w}$ to lie in the region $||\mathbf{w}||_1 \leq t$, the lowest achievable objective value, $\hat{\mathbf{w}}$, lies at the point of the feasible region, at which $w_1 = 0$ and $w_2 = t$.

The Lagrange Dual problem

Equation 5.6 may be written in *Lagrangian form*

$$\underset{\mathbf{w}\in\mathbb{R}^p}{\arg\min}\ \frac{1}{2}||\mathbf{x} - \mathbf{V}\mathbf{w}||_2^2 + \lambda||\mathbf{w}||_1 \tag{5.7}$$

where we have a different hyper-parameter controlling sparsity $\lambda > 0$. By Lagrangian duality [e.g., Bishop, 2006, Appendix E], for any value of t in the constrained form where the constraint is active, these exists a value of λ in the Lagrangian form such that the two problems have the same solution. One can also translate the other way; if $\hat{\mathbf{w}}_\lambda$ is a solution to the Lagrangian form, then it is also a solution to the constrained form with $t = ||\hat{\mathbf{w}}_\lambda||_1$.

It may be preferable to implement the Lasso by solving the Lagrangian form instead of the constrained form due to the lack of constraints. The Lagrange form also highlights a link to Bayesian inference. The function being minimised in **Equation 5.7** has the form of a negative log posterior with a Gaussian likelihood and i.i.d Laplace priors on $\mathbf{w}$. The squared error term is propor-

tional to $\prod_{i=1}^{n} \log \mathcal{N}(x_i|\mathbf{v}_i^\top \mathbf{w}, \sigma^2)$ for any σ^2, and the ℓ_1 term is proportional to $\log \prod_{i=1}^{k} \text{Laplace}(w_i|0, \frac{1}{\lambda})$. The evidence term does not appear, since it is constant in $\mathbf{w}$. Another way of obtaining the Lasso is thus to introduce Laplace priors on the parameters and perform MAP inference.

Computing the Lasso solution

The objective function in **Equation 5.7** is convex and – due to the ℓ_1 penalty – nondifferentiable. Following **Section 5.1.1**, to find the optimum $\mathbf{w}$ we require a point at which $\mathbf{0}$ is a subgradient, but this is not available in closed form. One way in which the Lasso solution can be computed is via *cyclic coordinate descent* [e.g., Hastie et al., 2015, Chapter 5], in which the objective is minimised with respect to each element of $\mathbf{w}$ in some fixed order until convergence.

To derive cyclic coordinate descent, we first rewrite the objective function in **Equation 5.7** as

$$f(\mathbf{w}) = \frac{1}{2}||\mathbf{r}^{(l)} - \mathbf{V}_{:,l} w_l||_2^2 + \lambda||\mathbf{w}_{-l}||_1 + \lambda|w_l| \tag{5.8}$$

Here we use $\mathbf{V}_{:,l}$ to indicate the l'th column of $\mathbf{V}$, and w_l to indicate l'th element of $\mathbf{w}$. $\mathbf{V}_{:,-l}$ and $\mathbf{w}_{-l}$ indicate $\mathbf{V}$ and $\mathbf{w}$ with their l'th row removed respectively. We have defined the *partial residual* $\mathbf{r}^{(l)} = \mathbf{x} - \mathbf{V}_{:,-l}\mathbf{w}_{-l}$ which is the residual from the current fit ignoring the l'th predictor.

We now compute the subdifferential

$$\frac{\partial f}{\partial w_l} = w_l||\mathbf{V}_{:,l}||_2^2 - \mathbf{r}^{(l)\top}\mathbf{V}_{:,l} + \lambda \,\text{sign}(w_l) \tag{5.9}$$

At the optimum value w_l^* this will contain 0, so we consider a single $z \in \text{sign}(w_l^*)$ and solve for w_l^*, arriving at

$$w_l^* = \frac{\mathbf{r}^{(l)\top}\mathbf{V}_{:,l} - \lambda z}{||\mathbf{V}_{:,l}||_2^2} \tag{5.10}$$

Since the denominator is positive, the numerator has the same sign as w_l^* and z, allowing us to consider three cases $w_l^* < 0$, $w_l^* > 0$ and $w_l^* = 0$.

$$w_l^* = \frac{1}{||\mathbf{V}_{:,l}||_2^2} \begin{cases} \mathbf{r}^{(l)\top}\mathbf{V}_{:,l} - \lambda & \text{if } \mathbf{r}^{(l)\top}\mathbf{V}_{:,l} > \lambda \\ 0 & \text{if } |\mathbf{r}^{(l)\top}\mathbf{V}_{:,l}| \leq \lambda \\ \mathbf{r}^{(l)\top}\mathbf{V}_{:,l} + \lambda & \text{if } \mathbf{r}^{(l)\top}\mathbf{V}_{:,l} < -\lambda \end{cases} \tag{5.11}$$

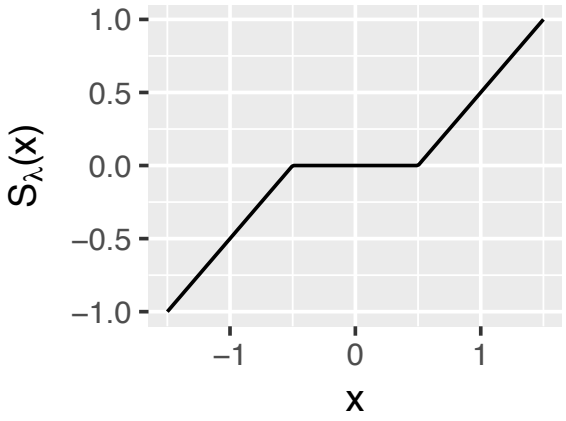


Figure 5.2: *The soft-thresholding operator using* $\lambda = 0.5$.

This can be written more compactly as

$$w_l^* = \frac{\mathcal{S}_\lambda(\mathbf{r}^{(l)\top}\mathbf{V}_{:,l})}{||\mathbf{V}_{:,l}||_2^2} \tag{5.12}$$

using the *soft-thresholding operator*, which we illustrate in **Figure 5.2**:

$$\mathcal{S}_\lambda(x) = \max(|x| - \lambda, 0)\,\text{sign}(x) \tag{5.13}$$

The elastic-net

The elastic-net [e.g., Hastie et al., 2015] is a generalisation of the Lasso, adding a combination of ridge and Lasso penalties:

$$\arg\min_{\mathbf{w}} \frac{1}{2}||\mathbf{y} - \mathbf{V}\mathbf{w}||_2^2 + \lambda\left[\frac{1-\alpha}{2}||\mathbf{w}||_2^2 + \alpha||\mathbf{w}||_1\right] \tag{5.14}$$

The parameter $\alpha \in [0, 1]$ smoothly interpolates between the Lasso ($\alpha = 1$) and ridge regression ($\alpha = 0$).

The Lasso can have erratic behaviour when variables are highly correlated (which is the case with FAIMS data). If input features $\mathbf{V}_{:,i}, \mathbf{V}_{:,j}$ are identical, then coefficients w_i and w_j are non-identifiable, and their learned values may depend upon initialisation and implementation. The addition of the ℓ_2 penalty recovers uniqueness since w_i, w_j will be given equal values.

The coordinate descent update step for the elastic-net has the form

$$w_l^* = \frac{\mathcal{S}_{\lambda\alpha}(\mathbf{r}^{(l)\top}\mathbf{V}_{:,l})}{||\mathbf{V}_{:,l}||_2^2 + \lambda(1-\alpha)} \tag{5.15}$$

which can be seen to be similar to the Lasso solution. The difference is in the reduced thresholding parameter $\lambda\alpha$, and with increased shrinkage due to the additional positive term $\lambda(1-\alpha)$ in the denominator.

Related models

The Lasso may be modified to achieve sparsity in different domains. So long as the regularisation term remains convex, the problem remains tractable.

One variant of the Lasso is used in Robust PCA (introduced in **Section 1.1.1**), which uses the nuclear norm in the penalty term to encourage an inferred matrix to be low-rank. The nuclear norm is the sum of the singular values of a matrix. Since these are positive for any real matrix, this is the same as the sum of the absolute singular values. This is similar to the Lasso, which uses a sum of absolute coefficients to achieve sparsity in the coefficients. By penalising the sum of the absolute singular values, we achieve sparsity in the singular values, and thus produce a low-rank matrix. Other variants can be found in Hastie et al. [2015, Chapter 4], which include the *group Lasso*, in which groups of variables are defined, and all variables in a group are set to zero together.

5.2 Model

SpStPCA is obtained by taking the structured Gaussian prior over $\mathbf{W}$ in StPCA and multiplying this by an i.i.d Laplace prior:

$$p(\theta|\beta, b) \propto \left[\prod_{i=1}^{k} \mathcal{N}(\mathbf{w}_i|\mathbf{0}, \mathbf{K}_\beta)\right]\left[\prod_{i=1}^{k}\prod_{j=1}^{d} \text{Laplace}(w_{ij}|0, b)\right] \tag{5.16}$$

The purpose of the Laplace prior is to introduce sparsity into the MAP $\mathbf{W}$. The amount of sparsity is modulated by the new hyper-parameter b, where smaller values of b lead to a sparser MAP $\mathbf{W}$. Introduction of this prior does not change the E-step in the EM procedure for MAP inference in StPCA. Only the M-step is affected, which we describe below.

The prior **Equation 5.16** does not integrate to 1, and there is no closed form normalising constant. This introduces an unknown constant into the posterior.

However we are primarily interested in finding the MAP, so this constant is not problematic.

5.2.1 Solving the SpStPCA MAP

The SpStPCA M-step is performed by maximising the expected complete log posterior

$$
\begin{aligned}
Q(\theta, \theta^{\text{old}}) &= \mathbb{E}\left[\log p(\mathbf{X}|\mathbf{V}, \theta)\right]_{p(\mathbf{V}|\mathbf{X}, \theta^{\text{old}})} + \log p(\theta|\beta, b) && (5.17)\\
&= -\frac{1}{2}\left(nd\log 2\pi + nd\log\sigma^2 + \sigma^{-2}\mathbb{E}\left[||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\text{F}}^2\right]\right) \\
&\quad -\frac{1}{2}\left(kd\log 2\pi + k\log|\mathbf{K}_\beta| + \text{Tr}\left[\mathbf{W}^\top\mathbf{K}_\beta^{-1}\mathbf{W}\right]\right) \\
&\quad -\left(\log 2b + \frac{1}{b}||\mathbf{W}||_1\right) && (5.18)
\end{aligned}
$$

This is the same as that for StPCA except with the additional term $-(\log 2b + \frac{1}{b}||\mathbf{W}||_1)$, which arises from the Laplace part of the SpStPCA prior.

Considering only the parameter $\mathbf{W}$, maximisation is performed by minimising the function

$$
\begin{aligned}
f(\mathbf{W}) :=& \frac{1}{2}\mathbb{E}\left[||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\text{F}}^2\right] + \sigma^2\left(\frac{1}{2}\text{Tr}\left[\mathbf{W}^\top\mathbf{K}_\beta^{-1}\mathbf{W}\right] + \frac{1}{b}||\mathbf{W}||_1\right) && (5.19)\\
=& \frac{1}{2}\mathbb{E}\left[||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\text{F}}^2\right] + \Omega(\mathbf{W}) && (5.20)\\
\propto& -Q(\theta, \theta^{\text{old}}) && (5.21)
\end{aligned}
$$

where we have introduced $\Omega(\mathbf{W})$ for cleaner syntax. In StPCA we derive a solution to the $\mathbf{W}$ maximisation step in terms of the solution to a Sylvester equation. In this case we are unable to obtain such a solution due to the additional term $||\mathbf{W}||_1$. We instead minimise **Equation 5.19** using coordinate descent. Below we derive an analytic minimiser for a single $w_{lm} \in \mathbf{W}$. The objective is convex, so cycling through each element of $\mathbf{W}$ and minimising converges at the global minimum.

Coordinate Descent

We can rewrite

$$\mathbf{X} - \mathbf{V}\mathbf{W}^\top = \mathbf{X} - \sum_{i=1}^{k} \mathbf{V}_{:,i}\mathbf{w}_i^\top \tag{5.22}$$

$$= \left(\mathbf{X} - \sum_{i\neq l}^{k} \mathbf{V}_{:,i}\mathbf{w}_i^\top\right) - \mathbf{V}_{:,l}\mathbf{w}_l^\top \tag{5.23}$$

$$= \mathbf{R}^l - \mathbf{V}_{:,l}\mathbf{w}_l^\top \tag{5.24}$$

where $\mathbf{V}_{:,l}$ is the column vector made from the l'th column of $\mathbf{V}$. We have introduced $\mathbf{R}^l$ which is the partial residual obtained by subtracting from the data the reconstructions made without the l'th latent dimension. Using **Equation 5.24** and the definition of the Frobenius norm, the squared reconstruction error may be written as

$$||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\mathrm{F}}^2 = \sum_{i=1}^{d} ||\mathbf{r}_i^l - \mathbf{V}_{:,l}w_{li}||_2^2 \tag{5.25}$$

where $\mathbf{r}_i^l$ is the i'th column of $\mathbf{R}^l$. This formulation is useful as it allows us to single out an element of $\mathbf{W}$, and enables us to rewrite $f(\mathbf{W})$ as

$$f(\mathbf{W}) = \frac{1}{2}\mathbb{E}\left[\sum_{i=1}^{d}\left(\mathbf{r}_i^{l\top}\mathbf{r}_i^l - 2w_{li}\mathbf{r}_i^{l\top}\mathbf{V}_{:,l} + w_{li}^2\mathbf{V}_{:,l}^\top\mathbf{V}_{:,l}\right)\right] + \Omega(\mathbf{W}) \tag{5.26}$$

$$= \frac{1}{2}\mathbb{E}\left[||\mathbf{R}^l||_{\mathrm{F}}^2\right] - \sum_{i=1}^{d} w_{li}\mathbb{E}\left[\mathbf{r}_i^{l\top}\mathbf{V}_{:,l}\right] + \frac{||\mathbf{w}_l||_2^2}{2}\mathrm{Tr}\left[\mathbb{E}\left[\mathbf{V}_{:,l}\mathbf{V}_{:,l}^\top\right]\right] + \Omega(\mathbf{W}) \tag{5.27}$$

To perform coordinate descent, we take the derivative with respect to a single element w_{lm}:

$$\frac{\partial f}{\partial w_{lm}} = w_{lm}\mathrm{Tr}\left[\mathbb{E}\left[\mathbf{V}_{:,l}\mathbf{V}_{:,l}^\top\right]\right] - \mathbb{E}\left[\mathbf{r}_m^{l\top}\mathbf{V}_{:,l}\right] + \frac{\partial\Omega}{\partial w_{lm}} \tag{5.28}$$

where

$$\frac{\partial\Omega}{\partial w_{lm}} = \sigma^2\left(\sum_{i=1}^{d} w_{li}(\mathbf{K}_\beta^{-1})_{m,i} + \frac{1}{b}\frac{\partial|w_{lm}|}{\partial w_{lm}}\right) \tag{5.29}$$

We now introduce z as an element of the subdifferential of $|w_{lm}|$ and solve

for the value w^*_{lm} at which $0 \in \frac{\partial f}{\partial w_{lm}}$, giving:

$$w^*_{lm} = \frac{\mathbb{E}\left[\mathbf{r}_m^{l\top}\mathbf{V}_{:,l}\right] - \sigma^2 \sum_{i\neq m}^{d} w_{li}(\mathbf{K}_\beta^{-1})_{m,i} - \frac{\sigma^2}{b}z}{\mathrm{Tr}\left[\mathbb{E}\left[\mathbf{V}_{:,l}\mathbf{V}_{:,l}^\top\right]\right] + \sigma^2(\mathbf{K}_\beta^{-1})_{m,m}} \tag{5.30}$$

Following the same process as the derivation for the Lasso (**Section 5.1.2**), this can be be written as

$$w^*_{lm} = \frac{\mathcal{S}_{\frac{\sigma^2}{b}}\left(\mathbb{E}\left[\mathbf{r}_m^{l\top}\mathbf{V}_{:,l}\right] - \sigma^2 \sum_{i\neq m}^{d} w_{li}(\mathbf{K}_\beta^{-1})_{m,i}\right)}{\mathrm{Tr}\left[\mathbb{E}\left[\mathbf{V}_{:,l}\mathbf{V}_{:,l}^\top\right]\right] + \sigma^2(\mathbf{K}_\beta^{-1})_{m,m}} \tag{5.31}$$

where $\mathcal{S}$ is the soft-thresholding operator introduced in **Section 5.1.2**. As expected, smaller values of b produce more thresholding. We also see more thresholding for larger values of σ^2. This makes sense as, if we believe the data are corrupted by a high level of noise, it takes a stronger signal to convince us that a given feature contains useful information.

Computing expectations

Two expectations must be computed to find w^*_{lm} in **Equation 5.31**, which we compute here. These are taken over $p(\mathbf{v}_i|\mathbf{x}_i, \theta)$, which in StPCA is

$$p(\mathbf{v}_i|\mathbf{x}_i, \theta) = \mathcal{N}(\mathbf{v}_i|\mathbf{M}^{-1}\mathbf{W}\mathbf{x}_i, \sigma^2\mathbf{M}^{-1}) \tag{5.32}$$

This remains true in SpStPCA since we have only altered the prior, which $p(\mathbf{v}_i|\mathbf{x}_i, \theta)$ does not depend on. Using this, we can obtain the following expression for the expectation

$$\mathbb{E}\left[\mathbf{V}_{:,l}\mathbf{V}_{:,l}^\top\right] = \mathrm{cov}(\mathbf{V}_{:,l}) + \mathbb{E}\left[\mathbf{V}_{:,l}\right]\mathbb{E}\left[\mathbf{V}_{:,l}\right]^\top \tag{5.33}$$

$$= \sigma^2(\mathbf{M}^{-1})_{l,l}I + (\mathbf{X}\mathbf{W}^\top\mathbf{M}^{-1})_{:,l}(\mathbf{X}\mathbf{W}^\top\mathbf{M}^{-1})_{:,l}^\top \tag{5.34}$$

The second expectation, $\mathbb{E}\left[\mathbf{r}_m^{l\top}\mathbf{V}_{:,l}\right]$, can be obtained by expanding out the definition of $\mathbf{r}_m^l$ and using independence to substitute $\mathbb{E}\left[\mathbf{V}_{:,i}^\top\mathbf{V}_{:,l}\right] = \mathbb{E}[\mathbf{V}_{:,i}]^\top\mathbb{E}[\mathbf{V}_{:,l}]$ for $i \neq l$, arriving at:

$$\mathbb{E}\left[\mathbf{r}_m^{l\top}\mathbf{V}_{:,l}\right] = \left(\mathbf{X}_{:,m} - \sum_{i\neq l}^{k} w_{im}\mathbb{E}[\mathbf{V}_{:,i}]\right)^\top\mathbb{E}[\mathbf{V}_{:,l}] \tag{5.35}$$

With these expectations, we can now perform cyclic coordinate descent to find the

MAP $\mathbf{W}$ in SpStPCA.

Relation to elastic-net

The M-step of SpStPCA is closely related to the elastic-net. If we choose $\mathbf{K}_\beta = (1 - \frac{1}{b})^{-1} I$ and substitute this into the function to be minimised during the $\mathbf{W}$-maximisation step (**Equation 5.19**), we obtain the minimisation problem

$$\arg\min_{\mathbf{W}} \frac{1}{2}\mathbb{E}\left[||\mathbf{X} - \mathbf{V}\mathbf{W}^\top||_{\mathrm{F}}^2\right] + \sigma^2\left(\frac{1 - b^{-1}}{2}||\mathbf{W}||_{\mathrm{F}}^2 + b^{-1}||\mathbf{W}||_1\right) \tag{5.36}$$

Comparing **Equation 5.36** to the elastic-net (**Equation 5.14**), we see see the problems are very similar. There is in fact an equivalence between the problems. Although the squared error term in **Equation 5.36** is under an expectation, this expectation is over $\mathbf{V}$ and the term remains quadratic in $\mathbf{W}$. Knowing this, we can see the equivalence with the hyper-parameter correspondence $\sigma^2 = \lambda$, $b^{-1} = \alpha$.

From this perspective, we can see SpStPCA as a generalisation of the elastic-net. In the elastic-net, all features are *a-priori* considered independent. SpStPCA generalises this by enabling features to be tied together in the prior through $\mathbf{K}_\beta$. By choosing $\mathbf{K}_\beta$ to be diagonal as above we return to considering independent features, so it is no surprise that we recover the elastic-net.

Mathematically, the SpStPCA maximisation step for $\mathbf{W}$ contains the term $\frac{1}{2}\operatorname{Tr}\left[\mathbf{W}^\top \mathbf{K}_\beta^{-1}\mathbf{W}\right]$, in which the elements of $\mathbf{W}$ interact through $\mathbf{K}_\beta$. When selecting $\mathbf{K}_\beta$ proportional to the identity matrix, this term collapses into a constant times $||\mathbf{W}||_{\mathrm{F}}^2$, as seen in **Equation 5.36**.

5.2.2 Consequences of sparsity prior

Posterior invariances

In StPCA the posterior is invariant to the transformation $\mathbf{W} \to \mathbf{W}\mathbf{R}$ for any orthonormal matrix $\mathbf{R}$. We use this in StPCA to rotate $\mathbf{W}$ such that each column points in an eigen-direction, improving interpretability. This is no-longer the case in SpStPCA as the prior does not have this rotational invariance, as such a transformation would affect the sparsity pattern, and, more generally, $||\mathbf{W}||_1 \neq ||\mathbf{W}\mathbf{R}||_1$.

The invariances the SpStPCA posterior does have are permutations in the columns of $\mathbf{W}$ and flipping the sign of any element of $\mathbf{W}$. In `stpca` we order the columns of $\mathbf{W}$ to be descending in magnitude, so column 1 still captures more variation than the remaining columns, though unlike StPCA it is not the direction of the most variance.

Evidence approximation

StPCA uses the Laplace approximation of the evidence, which requires finding the posterior mode and using the local curvature to approximate the entire posterior as an un-normalised Gaussian, then computing the normalising constant (the approximate evidence) in closed form. The Laplace approximation cannot be used for SpStPCA because the posterior is not differentiable anywhere any element of $\mathbf{W}$ is 0, which may be the case at the mode. The evidence approximation for StPCA thus cannot be used for SpStPCA.

Computing the entire b-path

It may be desirable to fit StPCA for a range of values of b to see how the sparsity pattern is affected, which we do in the experiments in the next section. This can be performed efficiently by starting from a large value of b, corresponding to non-sparse solutions, initialising from StPCA then fitting with the coordinate descent algorithm described above. We then adjust b to the second largest value and fit SpStPCA again, initialising from the previous fit.

Empirically this saves a large amount of computation compared to fitting for each value of b separately, allowing SpStPCA to cheaply be fitted over an entire range of values of b. This is the same procedure as *pathwise coordinate descent* as used in the Lasso to perform the same task of fitting over a range of values of the sparsity parameter [Hastie et al., 2015].

5.3 Results

In this section we use computational experiments to investigate the behaviour of SpStPCA. The dataset we use is a set of 64 FAIMS measurements of breath from 53 IBD patients and 11 healthy controls. This is the dataset described in **Section 2.3.2** and used in the StPCA computational experiments in **Section 4.2**.

As with the StPCA experiments, we use a subset of the FAIMS features to reduce the computational costs. To make the experiments comparable, we use the same feature subset of 330 features used in the **Section 4.2** experiments. We also use the same covariance function (noisy squared exponential), and the values β and k as selected in the StPCA experiment.

This section is split into three parts. In **Section 5.3.1**, we look at how SpStPCA performs in an unsupervised setting, and investigate how the value of b impacts the sparsity in the loadings. We then investigate the accuracy of dis-

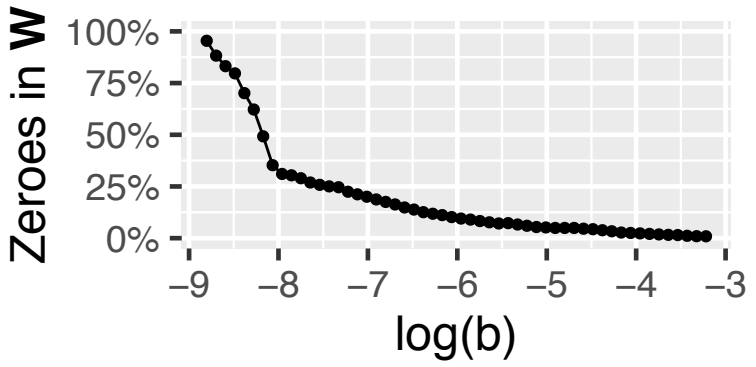


Figure 5.3: *As b is decreased, the number of zeroes in* $\mathbf{W}$ *increases.*

ease/control predictions made from the learned latent space in **Section 5.3.2**, and compare these to the results obtained by StPCA on the FAIMS data experiments in **Section 4.2**.

5.3.1 Effect of varying b

By fitting SpStPCA for a range of values of b and considering the number of zeroes in $\mathbf{W}$, we obtain **Figure 5.3**. This shows a monotonic increase in the sparsity of $\mathbf{W}$ as b is decreased, which is to be expected.

It is of interest to know whether, once a given element of $\mathbf{W}$ is set to zero, the element remains at zero for all smaller values of b. In **Figure 5.4** we plot which elements are zero for each value of b considered. In the left-hand plot the elements are presented in an arbitrary order. On the right-hand plot they are ordered by the total time spent at zero. This plot shows us that once an element of $\mathbf{W}$ becomes zero, it will often remain zero for all smaller values of b, but this is not always the case. This non-monotonicity of selection is also a property of SPCA [Hastie et al., 2015].

The relationship between coefficients being zero and the location of the coefficients $\mathbf{t}_i$ is also of interest. We illustrate this in **Figure 5.5** by plotting the values of $\mathbf{w}_1$, and whether they equal zero, at the locations $\mathbf{t}_i$. It can be seen that decreasing b causes all coefficients to shrink. However, those elements corresponding to pixels on the outside of the FAIMS plume are set to zero first.

Automatic model and feature selection

SpStPCA is able to perform automatic feature selection and automatic model selection for k. As $\mathbf{W}$ becomes more sparse, entire columns are set to zero. This is

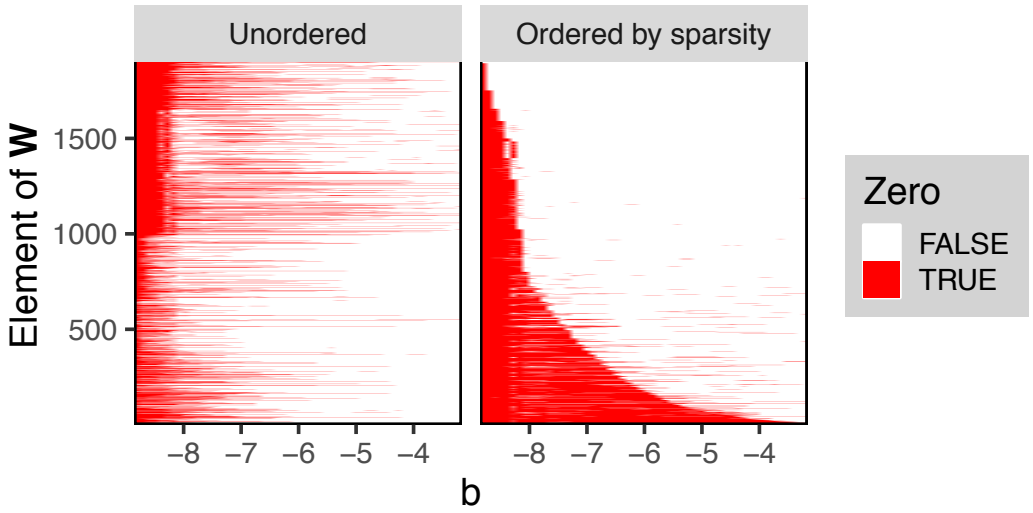


Figure 5.4: *Each vertical slice of this plot shows whether each element of* $\mathbf{W}$ *is zero at a given value of* b*. The difference between the two plots is the ordering of the elements of* $\mathbf{W}$*; the left-hand plot shows the stacked columns, and the right-hand plot has sorted the elements by the total number of 'false' values over the entire row. It can be seen that the selection of a feature is fairly stable; once a coefficient becomes zero, it usually remains zero. This is not always the case, and coefficients may switch between being zero and non-zero, though empirically rapid switching rarely occurs.*

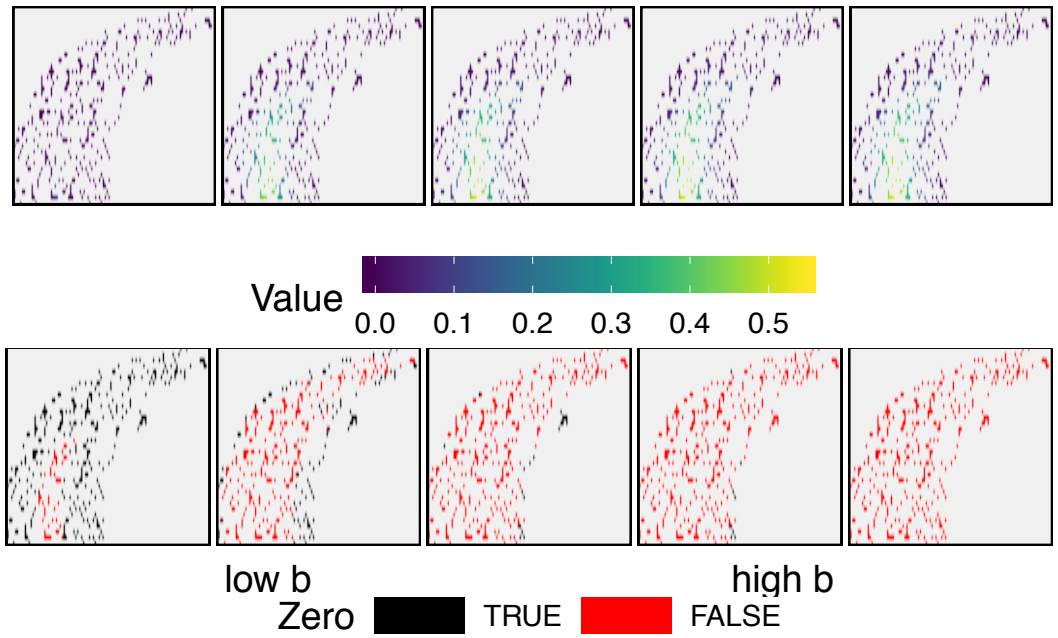
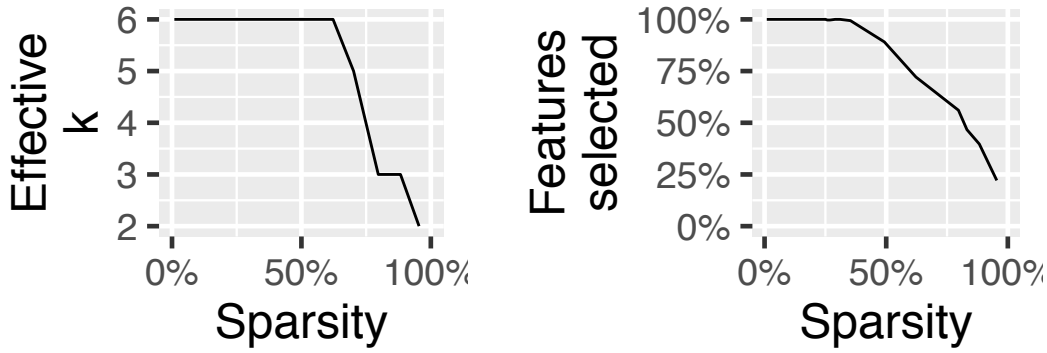


Figure 5.5: *Top: values in* $\mathbf{w}_1$ *plotted at their locations* $\mathbf{t}_i$*. Bottom: Indicator of whether the elements of* $\mathbf{w}_1$ *are zero (black) or non-zero (red). As we decrease* b *from high (right) to low (left), we can see shrinkage on the coefficients in the top row. We also see, in the bottom row, that the first features set to zero are those on the outside of the plume.*



(a) ***Model Selection:*** *When an entire column of* $\mathbf{W}$ *goes to zero, the effective value of* k *is reduced, thus automatic model selection is performed.*

(b) ***Feature Selection:*** *When an entire row of* $\mathbf{W}$ *is set to zero, the corresponding feature is unused. The unused set of features is thus determined automatically.*

Figure 5.6: *Automatic model and feature selection*

accompanied by the variance in the corresponding latent dimensions also shrinking to zero. When this occurs the effective value of k is reduced. Similarly, when an entire row of $\mathbf{W}$ becomes zero, the corresponding feature is no-longer used when computing the latent representations, thus selection of a subset of features to use is performed automatically. These are demonstrated in **Figure 5.6**, where we show how the percentage of features selected and the effective value of k change with the sparsity of $\mathbf{W}$.

5.3.2 Disease prediction

Experiments performed here are for the purpose of discovering whether the addition of sparsity aids in producing a latent space in which FAIMS data can be accurately classified.

In **Section 4.2**, StPCA was fitted to the 330-dimensional FAIMS data. In applying SpStPCA in this section, we take the model-selected StPCA values of k and β (using the SE covariance function), and use these in SpStPCA. We then fit SpStPCA with a range of values of b and see how the predictive performance changes. Predictive performance is measured using 10-fold cross-validated AUC, where predictions mare made using the Random Forest (RF) classifier.

The AUCs obtained are shown in **Figure 5.7**. For comparison, AUCs obtained using PPCA, SPCA and ICA are also included. For large b, SpStPCA obtains the same performance as StPCA in **Section 4.2**. This is unsurprising, as large b means little shrinkage, producing a model closer to StPCA. As b is reduced from

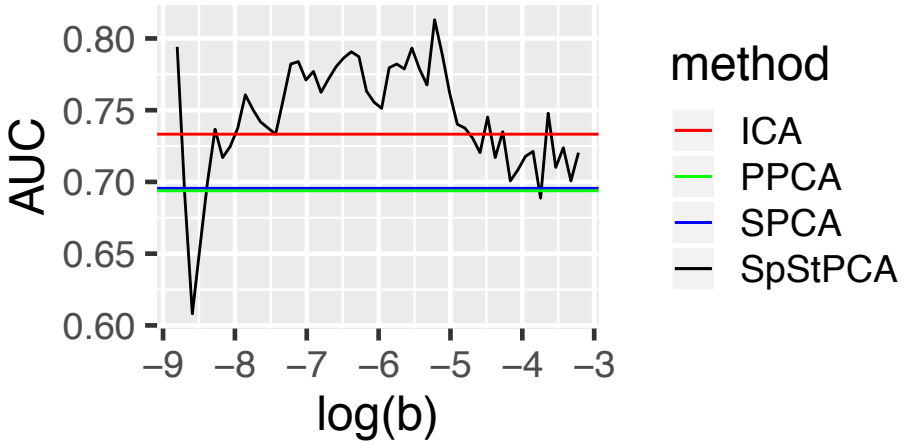


Figure 5.7: *10-fold cross-validated AUCs using Random Forest classifier. For large b the model becomes equal to StPCA. Greater predictive accuracy can be obtained by decreasing b, introducing sparsity in* $\mathbf{W}$.

the maximum value we see an increase in performance. This is interesting, since this corresponds to greater regularisation and suggests that StPCA was perhaps too flexible. Further reducing b gives a trough in performance, followed by another peak where only 73 features are used.

Comparison to SPCA

A fair comparison of the above test of SpStPCA is to SPCA, which also has a parameter we can alter to change the amount of sparsity in the loadings. We fit SPCA with $k = 6$ over a range of sparsities, and again compute 10-fold cross-validated AUCs, comparing the performance of SPCA and SpStPCA against the sparsity in the loadings. Results are shown in **Figure 5.8**, where SpStPCA outperforms SPCA over most of the sparsity range. The performance boost seen in SpStPCA must be due to the spatial prior, since this is the only way in which the models differ.

5.3.3 Generalising across datasets

Here we compare SpStPCA to SPCA in generalising across FAIMS disease classification tasks. We take the models fitted to IBD in the previous section and investigate how these fitted models generalise to the TB dataset used in **Section 2.3.3**. This TB dataset is chosen since, similarly to the IBD data, this dataset was collected using the FAIMS instrument, and is also of breath measurements. The TB data are reduced to the same set of 330 features as used in the previous section.

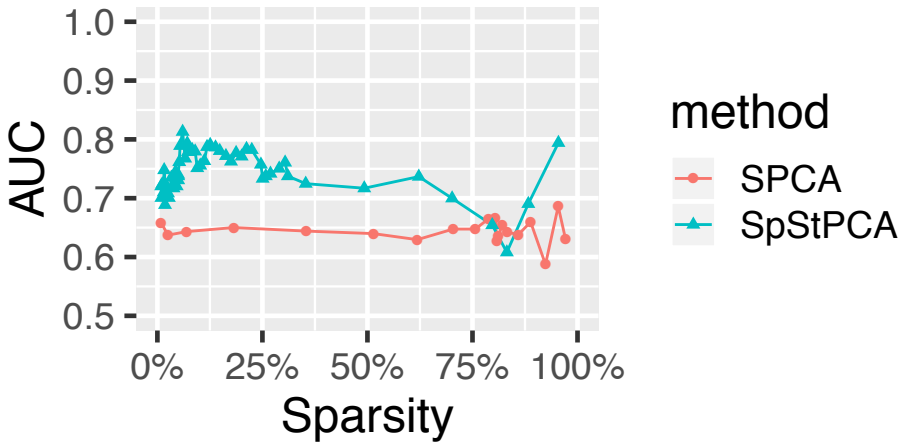


Figure 5.8: *Both SpStPCA and SPCA have a parameter controlling sparsity. By tweaking these and considering the predictive performance achieved and the level of sparsity in the loadings, the two methods can be compared. We can see that SpStPCA produces higher AUCs for nearly all sparsity levels considered. The difference in performance must be due to the spatial prior in SpStPCA, since this is the only way in which the methods differ.*

We first consider the generalisability of the parameters learned on the IBD data. Taking the SpStPCA and SPCA models achieving the greatest AUCs in **Figure 5.8**, we apply the feature transformations as learned on the IBD data to the TB data and evaluate predictive performance. As before, performance is measured using AUC produced in a 10-fold cross-validation using the Random Forest classifier. SpStPCA and SPCA obtain AUCs of 0.91 and 0.84 respectively.

We then investigate generalisability of the hyper-parameters. Considering the SpStPCA and SPCA hyper-parameter values which obtain the greatest predictive performance in **Figure 5.8**, we fit SpStPCA and SPCA to the TB data and again produce cross-validated AUCs. SpStPCA and SPCA obtain AUCs of 0.91 and 0.83 respectively.

Interestingly, the original study in **Section 2.3.3** obtained a leave-one-out cross-validated AUC of 0.83. SpStPCA out-performs this when generalising both the parameters and hyper-parameters from IBD, and SPCA performs similarly.

As a final experiment we fit SpStPCA and SPCA over a range of sparsity hyper-parameter values and investigate how their performance changes with the sparsity of the loadings. This produces the plot in **Figure 5.9**, in which we can see that for almost every given level of sparsity, SPCA out-performs SpStPCA. The reason we obtained greater AUCs for SpStPCA earlier in this section is that the hyper-parameters used from the IBD experiments produce different sparsity levels. SPCA produces close to 100% sparsity, whilst SpStPCA produces lower sparsity so

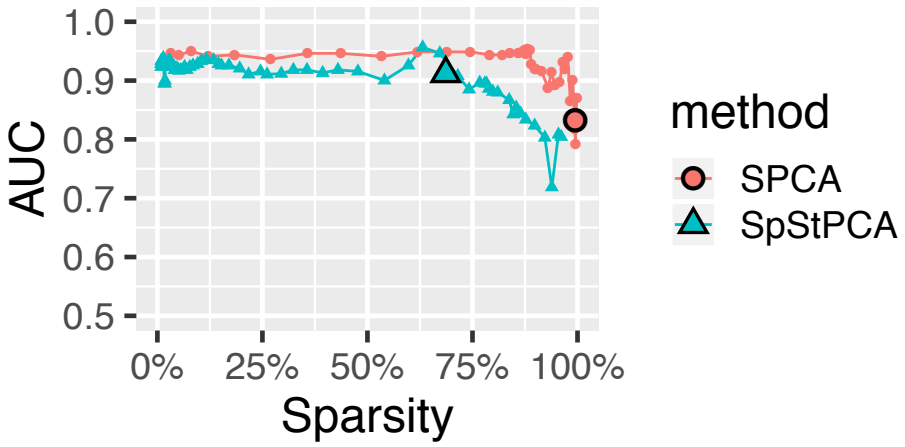


Figure 5.9: *SPCA seems to out-perform SpStPCA when considering the AUCs obtained over a range of sparsity levels. However, when taking the maximum AUC hyper-parameters from the IBD data and applying them to the TB data, SpStPCA selects a lower level of sparsity so obtains a greater AUC. The highlighted points show the AUCs achieved with the IBD selected hyper-parameters.*

obtains a higher AUC on the TB data.

The sparsity level for SpStPCA is better performing than that for SPCA, and also closer to the optimum for SpStPCA, showing better generalisation than the SPCA hyper-parameter. Note that if both techniques had picked their optimum sparsity levels they would have performed similarly, with SpStPCA and SPCA obtaining AUCs of 0.957 and 0.960 respectively.

5.4 Future Work

One drawback of SpStPCA is that the Laplace approximation to the marginal likelihood cannot be used. Due to the Laplace distribution term of the SpStPCA prior, the posterior contains nondifferentiable points which may be at the mode, meaning the Hessian is not defined and the Laplace approximation cannot be applied. For future work one could derive a marginal likelihood approximation which gets around this issue. We suggest a possible route here, inspired by the Laplace approximation. One could also investigate variational approximations.

We follow a similar process to the Laplace approximation, first defining vectors $\theta^{\text{z}} \subseteq \theta$ and $\theta^{\text{nz}} \subseteq \theta$ of the parameters with their MAP value at zero and non-zero respectively. We also define $|\theta^{\text{z}}|$ and $|\theta^{\text{nz}}|$ to be the length of each vector. First we

make an approximation to the un-normalised posterior:

$$p(\mathbf{X}|\theta)p(\theta|\beta,b) \approx p(\mathbf{X}|\hat{\theta})p(\hat{\theta}|\beta,b)\exp\left\{-\frac{1}{2}(\theta^{\mathrm{nz}}-\theta^{\hat{\mathrm{nz}}})^{\top}\mathbf{H}(\theta^{\mathrm{nz}}-\theta^{\hat{\mathrm{nz}}})-\sum_{i=1}^{|\theta^{\mathrm{z}}|}\frac{|\theta_i^{\mathrm{z}}|}{s_i}\right\} \tag{5.37}$$

$$\propto \mathcal{N}(\theta^{\mathrm{nz}}|\theta^{\hat{\mathrm{nz}}},\mathbf{H}^{-1})\prod_{i=1}^{|\theta^{\mathrm{z}}|}\mathrm{Laplace}(\theta_i^{\mathrm{z}}|0,s_i) \tag{5.38}$$

Here $\mathbf{H}^{-1}$ is the negative Hessian of the un-normalised log posterior with respect to θ^{nz}, as in the Laplace approximation. $\theta^{\hat{\mathrm{nz}}}$ is the MAP value of θ^{nz}; $\hat{\theta}^{\mathrm{z}}$ is the zero-vector, so is not included as a symbol in **Equation 5.37**. If all variables are non-zero at the MAP, θ^{z} is empty and the Laplace approximation is recovered exactly. There are $|\theta^{\mathrm{z}}|$ scale variables s_i that need to be determined. We know that the un-normalised posterior has a non-differentiable mode at 0 for each parameter in θ^{z}; the values of each s_i may be determined by matching the subgradients of the approximation to the subgradients of the un-normalised log posterior.

The marginal likelihood approximation is found by computing the area under **Equation 5.37**, which can be done in closed form. When integrating **Equation 5.37** over θ, this splits in to a product of a Gaussian integral over θ^{nz}, and $|\theta^{\mathrm{z}}|$ one-dimensional integrals over each θ_i^{z}. The resulting normalising constant is:

$$p(\mathbf{X}|\beta,b) \approx p(\mathbf{X}|\hat{\theta})p(\hat{\theta}|\beta,b)(2\pi)^{\frac{|\theta^{\mathrm{z}}|}{2}}|\mathbf{H}|^{-\frac{1}{2}}\prod_{i=1}^{|\theta^{\mathrm{z}}|}2s_i \tag{5.39}$$

5.5 Conclusion

SpStPCA is a linear latent variable model with a prior obtained by multiplying together the structured prior from StPCA and i.i.d Laplace priors. Introducing the Laplace priors means the MAP loadings are frequently sparse, which has the advantages of improved interpretability, as well as feature selection and model selection for the latent dimensionality.

SpStPCA generalises SPCA. In SPCA, the features are *a-priori* independent. In SpStPCA this is not the case due to the structured part of the prior, as in StPCA. Independence may be recovered by providing a diagonal covariance matrix to the structured prior. Furthermore, the SPCA model may be recovered exactly by correctly parametrising the covariance matrix.

SpStPCA also generalises StPCA. By choosing the sparsity hyper-parameter

b to be very large, corresponding to less sparsity, the SpStPCA MAP coincides with that of StPCA. SpStPCA may be efficiently fitted over a range of b values, enabling exploration.

We perform computational experiments, investigating how SpStPCA performs on FAIMS data compared to SPCA. Performance is measured via AUCs in classifying disease samples from healthy controls in 10-fold cross-validation. We first consider a dataset of breath measurements from IBD-positive patients and healthy controls. This is the same dataset used in the StPCA experiments, enabling comparison. For large b, SpStPCA performs the same as StPCA as expected. By decreasing b, and thus introducing sparsity, the AUCs obtained increase and out-perform all methods used in the StPCA experiments on the same dataset. Furthermore, when comparing SpStPCA to SPCA over a range of sparsity hyper-parameter values, SpStPCA produces greater AUCs over the majority of the range of possible sparsity values.

We then consider how SpStPCA generalises to a new dataset compared to SPCA. We consider the TB dataset from **Section 2.3.3**, and investigate how the transformations learned on the IBD dataset perform when applied to the TB dataset. We also investigate how the best performing hyper-parameter values on IBD generalise to TB. In both cases StPCA outperforms SPCA. This appears to be due to SPCA selecting too high a level of sparsity on IBD.

The Laplace approximation used in StPCA to approximate the evidence and posterior cannot be used in SpStPCA due to non-differentiable points at the MAP. However, we suggest an alternative evidence approximation.

Chapter 6

Conclusions

The devices studied here – inspired by olfaction, and in the early stages of medical use – are showing great promise. We have seen multiple studies where biological samples from hospital patients displaying some disease can be distinguished from that of healthy controls using an artificial olfaction instrument. We have also seen cases where disease subtypes have been distinguished between, easing some of our concerns that we are only picking up on a generic "poor health" signal. Human olfaction already plays a vital part of certain diagnostic procedures, and we have learned that canines, with their superior olfactory capabilities, can detect medically interesting signal beyond the reach of humans. There seems, to the author, to be a great deal of evidence that the odours given off by biological samples contain valuable information on the state of health. Until now this information source has been largely closed to us, but with the introduction of these devices, odour has become a phenomenon which can be measured and quantified.

Scientific studies performed for evaluating the medical potential of artificial olfaction instrumentation have typically only collected a small number of samples. This is understandable given their exploratory nature, but limits our ability to draw confident conclusions, especially when the signals are fairly weak. Compounding this, the data are typically high dimensional, and standard feature learning techniques such as PCA are generally applied. A problem here is that a statistical model of the data is re-learned on each analysis, producing high variance in the learned parameters. This means that a study may appear to fail due to learning a poor set of parameters, despite signal existing within the data. However, we typically have plenty of prior knowledge about the structure of the data, and utilising this would allow us to squeeze the best results out of the data we have.

In this thesis we have introduced the new statistical models StPCA and

SpStPCA, for which we have produced implementations[1]. These are linear latent variable models which allow a user to specify a prior over the covariance structure of the data, and thus to learn a more accurate model from fewer samples, provided the prior is good. This prior may be intuitively specified via a covariance function, and may help a researcher get the most out of a small quantity of data. StPCA allows an approximate marginal likelihood to be computed, leading to an approximate posterior and allowing a researcher to examine the certainty with which parameters have been learned. Principled Bayesian model selection is also enabled, allowing quantification of the support for one model (specified by latent dimensionality, covariance function and hyper-parameters) over another. SpStPCA extends StPCA by introducing sparsity on the parameter estimates, improving interpretability and empirically improving performance on FAIMS data. This extension unfortunately means the evidence approximation no longer applies, but we suggest a possible alternative.

StPCA and SpStPCA generalise ridge regression and the elastic net respectively, introducing correlation structure between the features. In both cases there is an ℓ_2 penalty term on the coefficients, independently encouraging coefficients to be small. In this work, the coefficients to which the ℓ_2 norm is applied are tied together through the prior covariance matrix. Coefficients which are described as a-priori more correlated are encouraged to have similar values. As a result, features in StPCA and SpStPCA are not necessarily a-priori independent.

StPCA and SpStPCA were designed for use with artificial olfaction data, but are highly general methods which are made application-specific via a user-specified covariance function in the prior. These should both be of use in any field where one wishes to fit a latent variable model and has prior knowledge of the covariance structure of the data. This would most likely be useful in cases of high dimension and small sample number. Data such as images or time-series, where we expect nearby points to take similar values, are clear candidates for application, but the full range of applications is very broad.

There is still a great deal more exploration and development to be done before these instruments can be routinely applied in medicine. Studies performing multi-class classification between a larger number of diseases and subtypes would be interesting, as we may discover that, although samples from a range of diseases can all be distinguished from control samples, the diseases themselves may not be distinguished between. An immediate thought here is to combine all datasets collected with a particular machine, but we have already seen that batch effects within

[1] `https://github.com/JimSkinner/stpca`

an experiment are common, so batch effects between experiments will likely dominate. These technologies are constructed to be sensitive to a broad array of signals, but this also means that many possible background signals may be confounding, making batch effects difficult to avoid. The measurements may be confounded by many signals irrelevant to health, such as sample storage time, differences in experimental procedure, patient diet and time since last meal. Careful study design is thus required to avoid accidental confounding of the data.

Interestingly, the BreathSpec project[2] analyses patient breath in-situ, eliminating storage time and experimental procedure as sources of variation, and thus as possible confounders. Another major source of variance lies in individual patient information, such as age or gender. Disease predictions typically come from the sample measurement alone, but including this patient information in the predictive model may be required to obtain the best predictive accuracy.

A likely application of value would be to discriminate between diseases which appear similar but demand different treatment, such as distinguishing bacterial from viral infections. In this thesis we have focused on the use of artificial olfaction in diagnosis, but this is unlikely to be the only medical application. Monitoring the progression or remission of a disease could be fruitful, especially since intra-patient variability will be less of an issue. However, given that artificial olfaction has the potential to provide a broadly sensitive, low-cost, non-invasive (depending on sample type), rapid and portable method of biomarker detection, it seems probable that there are many more medical uses which have not yet been considered.

What will the future of medicine making full use of artificial olfaction look like? This is clearly speculative, but one can imagine, due to their broad sensitivity, that a single artificial olfaction instrument can replace a large number of specialised tests, saving costs in terms of time, equipment and training. This may be of particular value in the developing world. The set of diseases and subtypes for which testing is cheap and easy may expand, and routine checks for common cancers may help early detection. There is still a large body of experimental work that must be done to reach these goals, and by building appropriate statistical tools we aim to get the most out of every experiment.

[2] `https://breathspec.com`

Appendices

Appendix A

Useful mathematical tools

A.1 Principal Angles

A useful tool to compare the "closeness" of two linear subspaces is to compute the Principal Angles between them [Knyazev and Zhu, 2012]. In this Thesis, Principal Angles have been computed using the R package `prangles` which has been written by the author and made available at `https://github.com/JimSkinner/prangles`. It is assumed that these subspaces contain the origin. For linear subspaces $\mathcal{X} \subset \mathbb{R}^n$ and $\mathcal{Y} \subset \mathbb{R}^n$ of dimension $p \leq n$ and $q \leq n$ respectively, the principal angles $\Theta(\mathcal{X}, \mathcal{Y})$ are a non-increasing list of angles of length $m := min(p, q)$.

A.1.1 Definition

Given two unit vectors $\mathbf{x}$ and $\mathbf{y}$, the acute angle between these vectors $\theta(\mathbf{x}, \mathbf{y})$ is defined as

$$\cos\theta(\mathbf{x}, \mathbf{y}) = |\mathbf{x}^\top \mathbf{y}| \tag{A.1}$$

This definition may be recursively extended to the Principal Angles $\Theta(\mathcal{X}, \mathcal{Y}) = [\theta_1, \cdots, \theta_m]$.

$$\cos(\theta_k) = \max_{\substack{\mathbf{x}\in\mathcal{X} \\ ||\mathbf{x}||=1}} \max_{\substack{\mathbf{y}\in\mathcal{Y} \\ ||\mathbf{y}||=1}} |\mathbf{x}^\top \mathbf{y}| = |\mathbf{x}_k^\top \mathbf{y}_k| \tag{A.2}$$

subject to the constraints

$$\mathbf{x}^\top \mathbf{x}_i = \mathbf{y}^\top \mathbf{y}_i = 0, \quad i = 1, \cdots, k-1 \tag{A.3}$$

So, $\mathbf{x}_1$ and $\mathbf{y}_1$ are only constrained to be unit vectors and are thus the most distant unit vectors in $\mathcal{X}$ and $\mathcal{Y}$ respectively. The corresponding θ_1 is the largest angle between the subspaces. $\mathbf{x}_2$ and $\mathbf{y}_2$ are then defined to be maximally distant

unit vectors, but constrained to be orthogonal to $\mathbf{x}_1$ and $\mathbf{y}_1$ respectively. Due to this constraint, it must be the case that $\theta_2 \leq \theta_1$.

A.1.2 Properties

If any of the angles are zero, the subspaces intersect. Since the subspaces contain the origin, this rules out the case of parallel subspaces, so any two subspaces intersect at at-least a point.

The number of principal angles equal to zero is the dimensionality of the space in which the subspaces intersect. For example, a single zero angle implies that the subspaces intersect along a line.

If the largest principal angle is zero, this means that one subspace is contained within the other. If the subspaces are of the same dimension, this means that they are the same subspace.

A.1.3 Computation

Here we consider how to compute the principal angles where we are given two matrices containing bases for subspaces $\mathbf{U} = [\mathbf{u}_1 \cdots \mathbf{u}_p] \in \mathbb{R}^{n \times p}$, $\mathbf{V} = [\mathbf{v}_1 \cdots \mathbf{v}_q] \in \mathbb{R}^{n \times q}$.

Efficient algorithms for computing the principal angles between the subspaces defined by $\mathbf{U}$ and $\mathbf{V}$ are described by Björck and Golub [1973]. A high level overview of such an algorithm is given in **Algorithm A.1**. First orthonormal bases $\mathbf{Q}_\mathrm{U}, \mathbf{Q}_\mathrm{V}$ for $\mathbf{U}, \mathbf{V}$ are extracted using QR decomposition. Then the singular values of $\mathbf{M} = \mathbf{Q}_\mathrm{U}^\top \mathbf{Q}_\mathrm{V}$ are computed, and the arccos of these values are the principal angles.

Algorithm A.1: Compute principal angles $\boldsymbol{\Theta}$ given bases $\mathbf{U}, \mathbf{V}$.

$\mathbf{Q}_\mathrm{U} \leftarrow \mathrm{QR}(\mathbf{U}).\mathrm{Q}$;
$\mathbf{Q}_\mathrm{V} \leftarrow \mathrm{QR}(\mathbf{V}).\mathrm{Q}$;
$\mathbf{M} \leftarrow \mathbf{Q}_\mathrm{U}^\top \mathbf{Q}_\mathrm{V}$;
$\mathbf{C} \leftarrow \mathrm{svd}(\mathbf{M}).\Sigma$;
$\boldsymbol{\Theta} \leftarrow [\arccos(\mathbf{C}_{ii}), \quad i \in 1, \cdots, m]$;
return $\boldsymbol{\Theta}$;

A.2 The single-element matix

The single-element matrix $\mathbf{J}_{ab}$ is a matrix of all zeroes except for the (a, b)'th element, which is a one. A comprehensive list of the properties of this matrix is given

by Petersen and Pedersen [2012, Section 9.7]. We list here only a few proprties which are used in this thesis.

The size of $\mathbf{J}_{ab}$ is defined such that any expression it is involved in is well defined. As used in this thesis it will typically be non-square.

A few properties of the single-element matrix are worth noting. Firstly, the single element matrix is an outer product of two single element vectors.

$$\mathbf{J}_{ab} = \mathbf{e}_a \mathbf{e}_b^\top \tag{A.4}$$

The product of two single-element matrices is either another single-element matrix or the matrix of zeroes, which can be seen easily noting that $\mathbf{e}_a^\top \mathbf{e}_b = \mathbb{1}[a = b]$.

$$\mathbf{J}_{ab}\mathbf{J}_{cd} = \mathbf{e}_a \mathbf{e}_b^\top \mathbf{e}_c \mathbf{e}_d^\top = \begin{cases} \mathbf{J}_{ad} & \text{if } b = c \\ \mathbf{0} & \text{if } b \neq c \end{cases} \tag{A.5}$$

Finally, two more complex relations:

$$\begin{aligned} \operatorname{Tr}[\mathbf{A}\mathbf{J}_{ab}] &= \operatorname{Tr}\left[\mathbf{e}_b^\top \mathbf{A} \mathbf{e}_a\right] \\ &= \operatorname{Tr}[\mathbf{A}_{ba}] \\ &= \mathbf{A}_{ba} \end{aligned} \tag{A.6}$$

$$\begin{aligned} \operatorname{Tr}[\mathbf{A}\mathbf{J}_{ab}\mathbf{B}\mathbf{J}_{cd}] &= \operatorname{Tr}\left[\mathbf{e}_d^\top \mathbf{A} \mathbf{e}_a \mathbf{e}_b^\top \mathbf{B} \mathbf{e}_c\right] \\ &= \mathbf{A}_{da}\mathbf{B}_{bc} \end{aligned} \tag{A.7}$$

A.3 Matrix derivitives

For a general matrix $\mathbf{Y}$ which is a function of x

$$\frac{\partial |\mathbf{Y}|}{\partial x} = |\mathbf{Y}| \operatorname{Tr}\left[\mathbf{Y}^{-1} \frac{\partial \mathbf{Y}}{\partial x}\right] \tag{A.8}$$

For invertible $\mathbf{Y}$ which is a function of x:

$$\frac{\partial \mathbf{Y}^{-1}}{\partial x} = -\mathbf{Y}^{-1} \frac{\partial \mathbf{Y}}{\partial x} \mathbf{Y}^{-1} \tag{A.9}$$

For any $\mathbf{Y}$:

$$\frac{\partial \operatorname{Tr}[\mathbf{Y}]}{\partial x} = \operatorname{Tr}\left[\frac{\partial [\mathbf{Y}]}{\partial x}\right] \tag{A.10}$$

which we can see because:

$$\frac{\partial \operatorname{Tr}[\mathbf{Y}]}{\partial x} = \frac{\partial}{\partial x} \sum_i \mathbf{Y}_{ii} \tag{A.11}$$

$$= \sum_i \frac{\partial}{\partial x} \mathbf{Y}_{ii} \tag{A.12}$$

$$= \operatorname{Tr}\left[\frac{\partial [\mathbf{Y}]}{\partial x}\right] \tag{A.13}$$

For general $\mathbf{X}$, $\mathbf{Y}$ both a function of x:

$$\frac{\partial \mathbf{X}\mathbf{Y}}{\partial x} = \mathbf{X}\frac{\partial \mathbf{Y}}{\partial x} + \frac{\partial \mathbf{X}}{\partial x}\mathbf{Y} \tag{A.14}$$

Appendix B

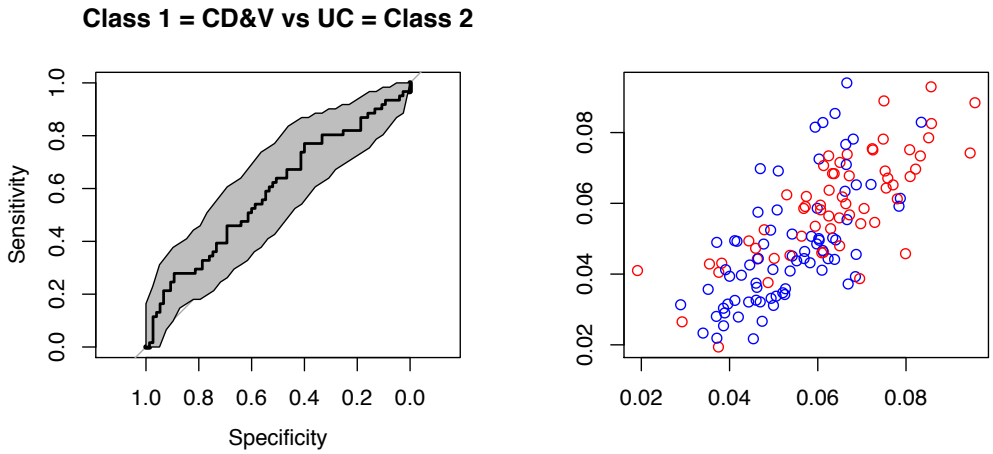
Analyses

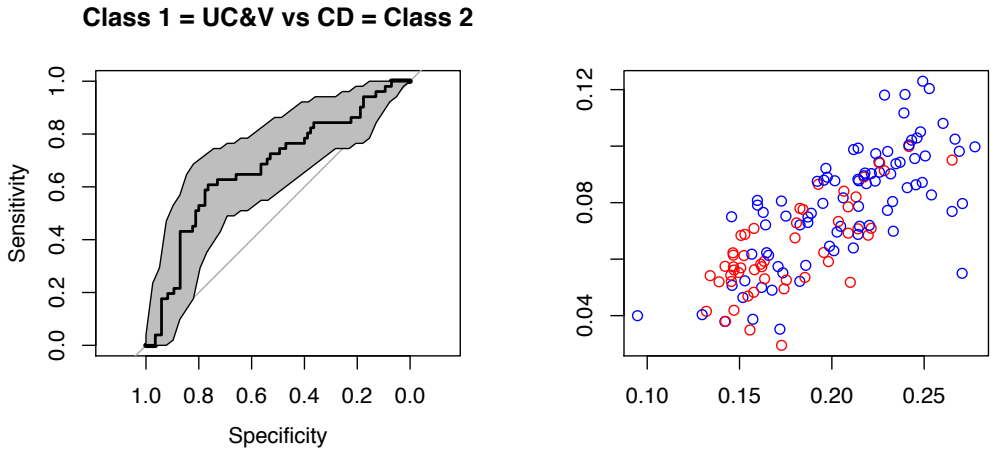
Here we list figures and results not directly applicable to the main thesis, but may be of interest to certain readers.

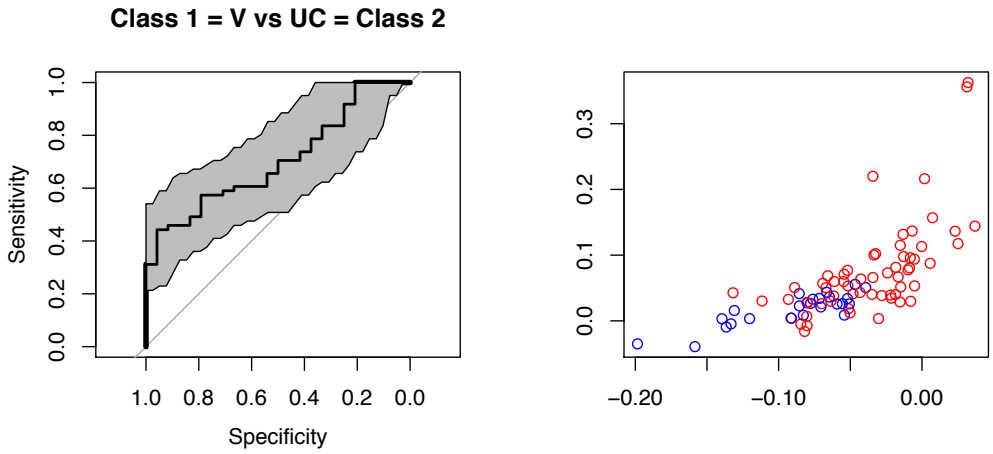
B.1 Non-invasive exhaled volatile organic biomarker analysis to detect Inflammatory Bowel Disease [Arasaradnam et al., 2016b]

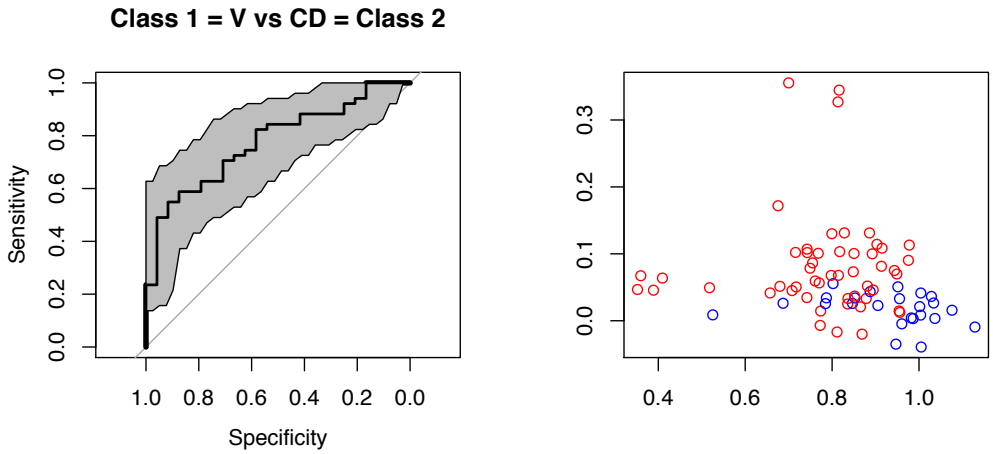
B.1.1 All classification tasks investigated

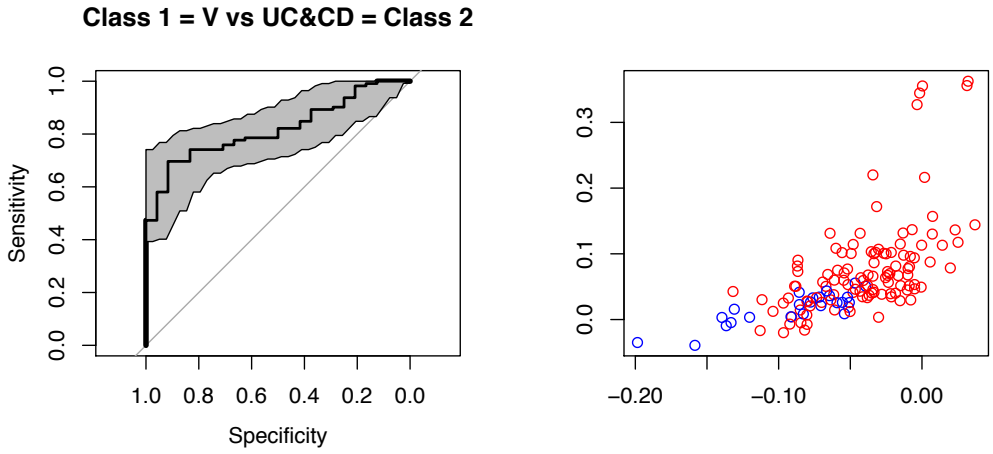
Here the ROC curves and AUCs for all classification tasks attempted are given. Since the pipeline reduces the data to two dimensions (by feature-selecting wavelets) before classification, we can plot the data in these two dimensions and see the separation of classes.

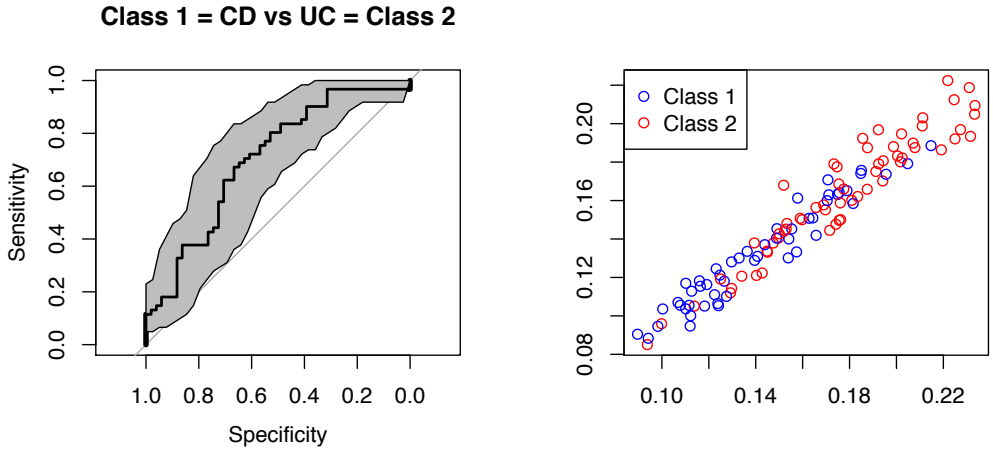

Class 1 = CD&V vs UC = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
0.02 0.04 0.06 0.08


Class 1 = UC&V vs CD = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
0.04 0.08 0.12
0.10 0.15 0.20 0.25


Class 1 = V vs UC = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.1 0.2 0.3
−0.20 −0.10 0.00


Class 1 = V vs CD = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.1 0.2 0.3
0.4 0.6 0.8 1.0


Class 1 = V vs UC&CD = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
0.0 0.1 0.2 0.3
−0.20 −0.10 0.00


Class 1 = CD vs UC = Class 2
Sensitivity
Specificity
1.0 0.8 0.6 0.4 0.2 0.0
0.0 0.2 0.4 0.6 0.8 1.0
Class 1
Class 2
0.08 0.12 0.16 0.20
0.10 0.14 0.18 0.22

B.1.2 AUCs obtained from run 1

The IBD analysis presented in the main body of the thesis uses only the second run of FAIMS measurements. Here we tabulate the results obtained using run 1.

Task	**AUC**	**95%CI**
UC vs CD,V	0.94	0.88 – 0.99
CD vs UC,V	0.63	0.49 – 0.77
UC vs V	0.68	0.47 – 0.89
CD vs V	0.51	0.31 – 0.71
UC,CD vs V	0.59	0.41 – 0.77
UC vs CD	0.61	0.45 – 0.77

Table B.1: *AUCs achieved using run 1 instead of 2*

Appendix C

R Package Documentation

R packages have been developed as part of this thesis. All R packages are hosted under the Github account of the author `https://github.com/JimSkinner`. Packages may be downloaded and installed simply with the `devtools` package. For example, to install the `gpclassifier` package one may enter the following into an R terminal.

```
library(devtools)
install_github("JimSkinner/gpclassifier")
```

C.1 gpclassifier

The R package `gpclassifier` (`https://github.com/JimSkinner/gpclassifier`) implements a Gaussian process Classifier for binary classification tasks. The implementation uses the Expectation Propagation approxmation, and was developed following the book *Gaussian Processes for Machine Learning* by Rasmussen and Williams [2005].

The most common usage pattern of the package would be to use the `GPC` function to train a classifier model on a matrix of training data `X.train` and training labels `Y.train`, and then `predict` a new set of labels on a set of test data `X.test`.

```
library(gpclassifier)
model.gp       = GPC(X.train, Y.train)
predictions.gp = predict(model.gp, X.text)
```

Package 'gpclassifier'

June 27, 2017

Type Package

Title Implements a Gaussian Process binary Classifier

Version 1.0

Date 2015-10-17

Author Jim Skinner

Maintainer Jim Skinner <j.r.skinner@warwick.ac.uk>

Description

R implementation of a Gaussian Process Classifier using the Expectation Propagation approximation detailed in (Gaussian Processes for Machine Learning; Rasmussen and Williams, 2006)

License MIT

Depends methods

Imports optimx, memoise, kernlab, functional

R topics documented:

covarFun *Construtor method of CovarFun class*

Description

Creates a new CovarFun object intended to be used inside a GPC Gaussian Process Classifier.

Usage

```
covarFun(k, dk, hp)
```

Arguments

k	A kernel function (object, x, y) -> numeric() which, given data x and y returns an inner product. Kernel hyperparameters may be accessed with object@hp.
dk	A function (object, x, y) -> list() which returns the gradient of k with respect to the hyprparameters in the form of a lsit of the same shape as the kyperparameters
hp	A list of kernel hyperparameters.

Details

A CovarFun object extends the kernel object which supplies a kernel function along with a list of hyperparameters. CovarFun also supplies a function returning the gradient of the kernel with respect to the hyperparameters, such that the hyperparameters may be tuned by the GPC class.

Examples

```
# Isotropic squared exponential covariance function with log length scale
# (ll) hyperparameter
library("gpclassifier")
k  = function(.Object, x, y) {
  exp(-0.5 * sum(exp(-2*.Object@hp$ll) * (x-y)^2))
}
hp = list(ll=0)
dk = function(.Object, x, y) {
  list(ll=.Object@k(.Object, x, y) * exp(-2*.Object@hp$ll)*crossprod(x-y))
}
C  = covarFun(k, dk, hp)
```

CovarFun-class *Class CovarFun.*

Description

Class CovarFun defines a covariance function to be used as part of a GPC gaussian process classifier. A CovarFun object is made up of a kernel k(x,y), a set of hyperparameters used in the kernel, and a function returning the gradient of the kernel with respect to each of the hyperparameters.

covarFun.LatentPlusNoise *Augment* CovarFun *with latent function and noise hyperparameters*

Description

Creates a covariance function k which augments some covariance function k_f with two extra hyperparameters lsf and lsn such that: k(x, y) = exp(lsf)*k_f(x, y) + exp(lsn)*I[x=y] Automatically provides derivatives of lsf, lsn. Used for neater covariance funtion specifications. lsf, lsn stand for log(sigma^2_f) and log(sigma^2_n), where 'f' labels the magnitude parameter and kernel for the latent function, whilst 'n' labels the magnitude parameter for the noise.

Usage

```
covarFun.LatentPlusNoise(covarFun)
```

Arguments

covarFun	CovarFun object to augment.

Value

covarFun augmented with lsf, lsn hyperparameterss for signal and noise magnitude.

Examples

```
C = covarFun.LatentPlusNoise(covarFun.SE(0))
```

covarFun.SE *Squared Exponential covariance function.*

Description

Squared Exponential covariance function.

Usage

```
covarFun.SE(ll = 0)
```

Arguments

ll	log length scale hyperparameter. ll must be of length 1 or the same length as the input vectors. If length 1 then this is an isotropic SE covar fun, else there is a length scale for each dimension of the input.

Value

CovarFun for a squared exponential covariance function

Examples

```
C <- covarFun.SE(0)
```

`EP` *Calculate site parameters, likelihood and likelihood gradient using Expectation Propagation*

Description

Calculate site parameters, likelihood and likelihood gradient using Expectation Propagation

Usage

```
EP(object)
```

`getCovarFun` *GPC Return the covariance function (class* `CovarFun`*) used.*

Description

GPC Return the covariance function (class `CovarFun`) used.

Usage

```
getCovarFun(object)
```

`getDLml` *Return partial derivatives of the log marginal likelihood with respect to each of the hyperparameters.*

Description

Return partial derivatives of the log marginal likelihood with respect to each of the hyperparameters.

Usage

```
getDLml(object)
```

`getHP` *Return the list of hyperparameters.*

Description

Return the list of hyperparameters.

Usage

```
getHP(object)
```

`getK`	*GPC Return the covariance matrix; the matrix of inner products between each pair of data points in the space induced by the covariance function.*

Description

GPC Return the covariance matrix; the matrix of inner products between each pair of data points in the space induced by the covariance function.

Usage

```
getK(object)
```

`getKernel`	*Return kernel function k:x,y -> numeric(). Differs from the kernel function specified when constructing the* `CovarFun`*, since the kernel function returned only requires parameters x,y, not object.*

Description

Return kernel function k:x,y -> numeric(). Differs from the kernel function specified when constructing the `CovarFun`, since the kernel function returned only requires parameters x,y, not object.

Usage

```
getKernel(object)
```

`getKernelGrad`	*Return kernel gradient function k:x,y -> list(). As with getKernel(CovarFun), this differs from the kernel gradient function specified when constructing the* `CovarFun`*, since the function returned only requires parameters x,y, not object.*

Description

Return kernel gradient function k:x,y -> list(). As with getKernel(CovarFun), this differs from the kernel gradient function specified when constructing the `CovarFun`, since the function returned only requires parameters x,y, not object.

Usage

```
getKernelGrad(object)
```

getLml *Return log marginal likelihood.*

Description

Return log marginal likelihood.

Usage

```
getLml(object)
```

GPC *Class GPC.*

Description

Class GPC defines a Gaussian Process Classifier. GPC(), GPC(X, Y, covarFun) creates a new GPC object used to predict labels for new input data.

Construct a new GPC object.

Usage

```
GPC(X, Y, covarFun = NA)

GPC(X, Y, covarFun = NA)
```

Arguments

X	Matrix of input data; sample in rows.
Y	Logical vector of binary labels.
covarFun	Covariance function to use. Must be of class CovarFun. If omitted, the squared exponential covariance function is used by default.

Details

Covariance function hyperparameters are selected automatically through maximum likelihood. This class is implemented using the Expectation Propagation approximation detailed in (Gaussian Processes for Machine Learning, Rasmussen and Williams, 2006).

Value

S4 object of class GPC, where covarance function hyperparameters have been set to their maximum likelihood estimates.

Examples

```
# Create synthetic dataset
X <- matrix(rnorm(60), ncol=2)
Y <- rowSums(X^2) < 1

# New GPX Object with default squared exponential covariance function.
gpc <- GPC(X, Y)

# Predict labels for new data
Xst <- matrix(rnorm(60), ncol=2)
Yst <- predict(gpc, Xst)
```

---

hpTune *Return maximum likelihood covariance function hyperparameters*

Description

Return maximum likelihood covariance function hyperparameters

Usage

```
hpTune(object)
```

---

predict,GPC-method *Predict labels from unlabelled data.*

Description

Use a trained GPC classifier to produce label predictions of a new set of unlabelled data.

Usage

```
## S4 method for signature 'GPC'
predict(object, Xst)
```

Arguments

object	A fitted GPC objet
Xst	A matrix of data for which to produce label predictions

Examples

```
# Create synthetic dataset
X <- matrix(rnorm(60), ncol=2)
Y <- rowSums(X^2) < 1

# New GPX Object with default squared exponential covariance function.
gpc <- GPC(X, Y)

# Predict labels for new data
Xst <- matrix(rnorm(60), ncol=2)
Yst <- predict(gpc, Xst)
```

setHP<- *Change the* CovarFun *hyperparameter list to the list supplied.*

Description

Change the CovarFun hyperparameter list to the list supplied.

Usage

```
setHP(object) <- value
```

update<- *Update* X, Y *or* covarFun

Description

Update the data X, labels Y or covariance function covarFun, causing a recalculation of hyperparameters, site parameters and covariance matrix.

Usage

```
update(object) <- value
```

C.2 `stpca`

The `stpca` R package (`https://github.com/JimSkinner/stpca`) implements Structured PCA as described in **Chapter 3**.

Package 'spca'

June 26, 2017

Title Structured PCA

Version 0.1

Author Jim Skinner <j.r.skinner@warwick.ac.uk>

Maintainer Jim Skinner <j.r.skinner@warwick.ac.uk>

Description R package for SPCA. A linear latent variable model similar to PPCA but with a user-specified prior on the covariance structure of each column of the loadings matrix. Supports tools for model selection.

Depends R (>= 3.1.0)

License MIT + file LICENSE

LazyData true

Collate 'synthesize-data.R'
'util.R'
'covariance-functions.R'
'evidence-approximation.R'
'spca.R'

Imports functional, MASS, fields, gsl, Matrix, numDeriv

RoxygenNote 6.0.1.9000

R topics documented:

`cov.RQ` *Rational quadratic covariance function*

Description

Rational quadratic covariance function

Usage

```
cov.RQ(X, X2, beta, D = NA, ...)
```

Examples

```
locations = matrix(rnorm(10), ncol=2)
beta      = rnorm(3) # Logarithms of variance, length scale & alpha

K = cov.RQ(locations, beta=beta)
stopifnot(all(Matrix::diag(K)==exp(beta[1]))) # Diagonal is exp(beta[1])
stopifnot(all(svd(K)$d>0)) # K is positive definite
stopifnot(all(K[upper.tri(K)]<K[1,1])) # Largest element is on diagonal
```

`cov.RQ.beta0` *Computationally cheap estimate for beta0 for cov.RQ.*

Description

Computationally cheap estimate for beta0 for cov.RQ.

Usage

```
cov.RQ.beta0(X, locations, k)
```

Examples

```
library(functional)
n = 10; k = 4; dim=c(10, 10); kern=Curry(cov.SE, beta=log(c(2, 0.4, 0.3)))
synth = synthesize_data_kern(n, k, dim, kern, noisesd=0.2)
beta0 = cov.RQ.beta0(synth$X, synth$grid, k)
stopifnot(all(is.finite(beta0)))
```

cov.RQ.d — *Rational quadratic covariance function derivatives wrt hyperparameters*

Description

Rational quadratic covariance function derivatives wrt hyperparameters

Usage

```
cov.RQ.d(X, X2, beta, D = NA, ...)
```

Examples

```
point1 = matrix(rnorm(1), ncol=1)
point2 = matrix(rnorm(1), ncol=1)
beta   = rnorm(3) # Logarithms of variance, length scale & alpha

library(numDeriv)
Ks.2points = vapply(cov.RQ.d(point1, point2, beta=beta), function(K) {
  K[1,1]
}, numeric(1))
Ks.2points.num = grad(function(beta_) {
  as.numeric(cov.RQ(point1, point2, beta=beta_))
}, x=beta)
stopifnot(all.equal(Ks.2points, Ks.2points.num))
```

cov.SE — *Squared exponential covariance function*

Description

Squared exponential covariance function

Usage

```
cov.SE(X, X2, beta, D = NA, ...)
```

Examples

```
grid = matrix(1:10, ncol=1)
beta = rnorm(2)
K    = cov.SE(grid, beta=beta)
stopifnot(all(Matrix::diag(K)==exp(beta[1])))
```

cov.SE.beta0 *Computationally cheap estimate for beta0 for cov.SE.*

Description

Computationally cheap estimate for beta0 for cov.SE.

Usage

```
cov.SE.beta0(X, locations, k)
```

Examples

```
library(functional)
n = 10; k = 4; dim=c(10, 10); kern=Curry(cov.SE, beta=log(c(2, 0.4)))
synth = synthesize_data_kern(n, k, dim, kern, noisesd=0.2)
beta0 = cov.SE.beta0(synth$X, synth$grid, k)
stopifnot(all(is.finite(beta0)))
```

cov.SE.d *Squared exponential covariance function derivatives wrt hyperparameters*

Description

Squared exponential covariance function derivatives wrt hyperparameters

Usage

```
cov.SE.d(X, X2, beta, D = NA, ...)
```

Examples

```
point1 = matrix(rnorm(1), ncol=1)
point2 = matrix(rnorm(1), ncol=1)
beta   = rnorm(2) # Logarithms of variance and length scale

Ks.1point = cov.SE.d(point1, beta=beta)

# Derivative wrt variance at zero distance should always be exp(beta[1])
stopifnot(all.equal(Ks.1point[[1]][1,1], exp(beta[1])))

# Derivative wrt lengthscale at zero distance should always be 0
stopifnot(all.equal(Ks.1point[[2]][1,1], 0))

# Identical tests with numerical gradient
library(numDeriv)
Ks.1point.num = grad(function(beta_) {
  cov.SE(point1, beta=beta_)[1,1]
}, x=beta)
stopifnot(all.equal(Ks.1point.num[1], exp(beta[1])))
stopifnot(all.equal(Ks.1point.num[2], 0))
```

```
Ks.2points = cov.SE.d(point1, point2, beta=beta)
Ks.2points.num = grad(function(beta_) {
  as.numeric(cov.SE(point1, point2, beta=beta_))
}, x=beta)

# Check numerical gradient equals analytic gradient
stopifnot(all.equal(as.numeric(Ks.2points[[1]]), Ks.2points.num[1]))
stopifnot(all.equal(as.numeric(Ks.2points[[2]]), Ks.2points.num[2]))
```

`min.f.generator` *Value and gradient of function to be minimized in tuning beta*

Description

Value and gradient of function to be minimized in tuning beta

Usage

```
## S3 method for class 'f.generator'
min(X, W, mu, sigSq, locations, covar.fn, covar.fn.d,
  D = NA, max.dist = Inf, sparse = FALSE)
```

Value

A function which, when given a value of beta, returns a value and gradient of a function to be minimized in beta-tuning.

Examples

```
set.seed(1)
n = 100
d = 30
k = 3
X = matrix(rnorm(n*d), ncol=d)
W = matrix(rnorm(d*k), nrow=d, ncol=k)
mu = rnorm(d)
sigSq = rnorm(1)^2
locations = matrix(rnorm(d*2), ncol=2)
beta = rnorm(2)

fdf = spca:::min.f.generator(X, W, mu, sigSq, locations, cov.SE, cov.SE.d)

library(numDeriv)
grad.analytic = fdf(beta)$df
grad.numeric = grad(function(beta_) {
  fdf(beta_)$f
}, x=beta)
all.equal(grad.analytic, grad.numeric, tolerance=100*.Machine$double.eps^0.5)
```

predict.spca *De-noise a sample using a trained* spca *object.*

Description

De-noise a sample using a trained spca object.

Usage

```
## S3 method for class 'spca'
predict(object, ...)
```

Arguments

object	An spca object returned from a call to spca
samples	samples to de-noise

Value

De-noised samples of the same dimensionality as the parameter samples

spca *Performs SPCA*

Description

Performs SPCA

Usage

```
spca(X, k, locations, covar.fn, covar.fn.d = NULL, beta0 = c(), trace = 0,
  report_iter = 10, max.dist = Inf, maxit = 20, maxit.outer = 5)
```

Arguments

X	Data
k	Latent dimensionality
locations	the coordinates of each dimenion in X
covar.fn	covariance function to generate K_beta
covar.fn.d	gradient of the covariance function with respect to hyperparameters
beta0	initial hyperparameters
trace	amount of reporting. 0=none, 1=low, 2=high
report_iter	Number of iterations between reports
max.dist	Maximum distance between features to consider
maxit	number of inner iterations
maxit.outer	number of outer iterations

Value

An spca object.

Examples

```
library(fields)
data(ozone2)

# Missing data: Replace missing values by column means
X = ozone2$y
for (col in 1:ncol(X)) {
  ind = is.na(X[,col])
  X[ind,col] = mean(X[,col], na.rm=TRUE)
}
X = X/sd(X) # Scale for numerical reasons

locations = ozone2$lon.lat
locations = apply(locations, 2, function(col) (col-min(col))/(max(col)-min(col)))

model.spca = spca(X, 3, locations, cov.SE, cov.SE.d, beta0=log(c(1, 0.5)),
                  maxit=20, maxit.outer=3, trace=0)
```

---

spca.H *Compute all the blocks of H.*

Description

Compute all the blocks of H.

Usage

```
spca.H(X, W, mu, sigSq, K)
```

Arguments

X	Data
W	Loadings matrix
K	Prior covariance matrix

Value

H

Examples

```
set.seed(1)
d=10; k=3; n=1000
X = matrix(rnorm(n*d), ncol=d)
W = matrix(rnorm(d*k), ncol=k)
mu = rnorm(d)
sigSq = rnorm(1)^2
K = cov.SE(matrix(1:10, ncol=1), beta=log(c(2, 3)))
```

```
library(numDeriv)
library(Matrix)

#Test that the analytic hessian for mu & sigSq matches numerical Hessian.
H.analytic = spca:::spca.H(X, W, mu, sigSq, K)
HsigSq.numeric = Matrix(hessian(function(sigSq_) {
  -spca.log_posterior(X, K, W, mu, sigSq_)
}, x=sigSq))
stopifnot(all.equal(H.analytic$sigSq, HsigSq.numeric,
                    tolerance=1e-8))

Hmu.numeric = Matrix(hessian(function(mu_) {
  -spca.log_posterior(X, K, W, mu_, sigSq)
}, x=mu))
stopifnot(all.equal(H.analytic$mu, Hmu.numeric, tolerance=1e-6))
```

`spca.H.W` *Compute all the w_i blocks of H*

Description

Compute all the w_i blocks of H

Usage

```
spca.H.W(X, W, mu, sigSq, K)
```

Arguments

`X`	Data
`W`	Loadings matrix
`K`	Prior covariance matrix

Value

H_w_i

Examples

```
set.seed(1)
d=10; k=3; n=10
X = matrix(rnorm(n*d), ncol=d)
W = matrix(rnorm(d*k), ncol=k)
mu = rnorm(d)
sigSq = rnorm(1)^2
K = cov.SE(matrix(1:10, ncol=1), beta=log(c(1, 3)))

library(numDeriv)
library(Matrix)
Hw1.analytic = spca:::spca.H.W(X, W, mu, sigSq, K)[[1]]
Hw1.numeric  = Matrix(hessian(function(w) {
  W_ = W
  W_[,1] = w
```

```
  -spca.log_posterior(X, K, W_, mu, sigSq)
}, x=W[,1]))

stopifnot(all.equal(Hw1.analytic, Hw1.numeric))
```

`spca.log_bayes_factor` *Compute bayes factor*

Description

Compute bayes factor

Usage

```
spca.log_bayes_factor(X, K1, W1, mu1, sigSq1, K2, W2, mu2, sigSq2)
```

Arguments

`X`	Data

Value

Log bayes factor; model 1 is numerator

`spca.log_evidence` *Compute the laplace approximation to the log evidence given the MAP parameters K, mu, sigSq as well as the prior covariance matrix K. Note that this is multiplied by an UN-KNOWN CONSTANT due to the flat priors over mu and sigSq. However, this unknown constant is always the same regardless of k and K, so this may be used to compute meaningful bayes factors between SPCA models.*

Description

Compute the laplace approximation to the log evidence given the MAP parameters K, mu, sigSq as well as the prior covariance matrix K. Note that this is multiplied by an UN-KNOWN CONSTANT due to the flat priors over mu and sigSq. However, this unknown constant is always the same regardless of k and K, so this may be used to compute meaningful bayes factors between SPCA models.

Usage

```
spca.log_evidence(X, K, W, mu, sigSq)
```

Arguments

`X`	Data
`K`	Prior covariance matrix
`W`	Loadings matrix

Value

Approximate log evidence

`spca.log_likelihood` *Does not yet work with 'new' SPCA architecture Calculate the log likelihood for SPCA with given parameters*

Description

Does not yet work with 'new' SPCA architecture Calculate the log likelihood for SPCA with given parameters

Usage

```
spca.log_likelihood(X, W, mu, sigSq)
```

Arguments

`X`	Data

Value

log likelihood (numeric)

Examples

```
d = 50
k = 5
n = 15

set.seed(1)
X  = matrix(rnorm(n*d), nrow=n, ncol=d)
mu = colMeans(X)
Xc = sweep(X, 2, mu, '-')
covar.svd = svd(Xc/sqrt(n), nu=0, nv=k)
covar.eigval = covar.svd$d^2
sigSq = sum(covar.eigval[-(1:k)])/(d-k)
W     = covar.svd$v %*% diag(sqrt(covar.eigval[1:k] - sigSq), ncol=k, nrow=k)

R     = svd(matrix(rnorm(k*k), ncol=k, nrow=k))$u # Random orthonormal matrix

# The likelihood is invariant to multiplying  by an orthonormal matrix.
l1 = spca.log_likelihood(X, W, mu, sigSq)
l2 = spca.log_likelihood(X, W%*%R, mu, sigSq)
stopifnot(all.equal(l1, l2))
```

spca.log_posterior *Calculate the *un-normalised* log posterior for SPCA with given parameters*

Description

Calculate the *un-normalised* log posterior for SPCA with given parameters

Usage

```
spca.log_posterior(X, K, W, mu, sigSq)
```

Arguments

X	Data
K	Prior covariance matrix
W	Loadings matrix

Value

un-normalised log posterior (numeric)

spca.log_prior *Calculate the *un-normalised* log prior for SPCA with given loadings matrix*

Description

Calculate the *un-normalised* log prior for SPCA with given loadings matrix

Usage

```
spca.log_prior(K, W)
```

Arguments

K	Prior covariance matrix
W	Loadings matrix

Value

un-normalised log prior (numeric)

synthesize_data *Synthesize fake data from SPCA model*

Description

Synthesize fake data from SPCA model

Usage

```
synthesize_data(n, k, K, noisesd = 0)
```

Arguments

n	Number of samples to create
k	Latent dimensionality
K	prior covariance matrix
noisesd	standard deviation of noise added to data

synthesize_data_kern *Synthesize fake data from SPCA model*

Description

Synthesize fake data from SPCA model

Usage

```
synthesize_data_kern(n, k, dim, kern, noisesd = 0)
```

Arguments

n	Number of samples to create
k	Latent dimensionality
dim	dimensions of grid to construct
kern	cvarinace function used to build the prior covariance matrix K
noisesd	standard deviation of noise added to data

Appendix D

StPCA

A number of derivations were ommitted from the main text body, and are instead presented here for the interested reader.

D.1 Model

D.1.1 Mean and covariance of the likelihood

In **Section 3.1.1**, the mean and covariance of the likelihood are stated but no detailed proof is given. The detailed proof is presented here. First we show the mean of $p(\mathbf{x}_i|\mathbf{W}, \sigma^2)$ is $\mathbf{0}$, which we do by computing the expected value.

$$\begin{aligned}
\mathbb{E}\big[\mathbf{x}_i|\mathbf{W}, \sigma^2\big] &= \int_{\mathbb{R}^d} \mathbf{x}_i p(\mathbf{x}_i|\mathbf{W}, \sigma^2)\, \mathrm{d}\mathbf{x}_i && \text{(D.1)}\\
&= \int_{\mathbb{R}^d} \mathbf{x}_i \int_{\mathbb{R}^k} p(\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2) p(\mathbf{v}_i)\, \mathrm{d}\mathbf{v}_i \mathrm{d}\mathbf{x}_i && \text{(D.2)}\\
&= \int_{\mathbb{R}^k} \left[\int_{\mathbb{R}^d} \mathbf{x}_i p(\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2)\, \mathrm{d}\mathbf{x}_i \right] p(\mathbf{v}_i)\, \mathrm{d}\mathbf{v}_i && \text{(D.3)}\\
&= \mathbb{E}\big[\mathbb{E}\big[\mathbf{x}_i|\mathbf{v}_i, \mathbf{W}, \sigma^2\big]\big] && \text{(D.4)}\\
&= \mathbb{E}\big[\mathbf{W}\mathbf{v}_i\big] && \text{(D.5)}\\
&= \mathbf{W}\mathbb{E}\big[\mathbf{v}_i\big] && \text{(D.6)}\\
&= \mathbf{0} && \text{(D.7)}
\end{aligned}$$

Now, using $\mathbb{E}\big[\mathbf{x}_i|\mathbf{W},\sigma^2\big]=\mathbf{0}$, we compute the covariance of $p(\mathbf{x}_i|\mathbf{W},\sigma^2)$:

$$\begin{aligned}
\operatorname{cov}(\mathbf{x}_i|\mathbf{W},\sigma^2) &= \mathbb{E}\big[\mathbf{x}_i\mathbf{x}_i^\top|\mathbf{W},\sigma^2\big]-\mathbb{E}\big[\mathbf{x}_i|\mathbf{W},\sigma^2\big]\mathbb{E}\big[\mathbf{x}_i|\mathbf{W},\sigma^2\big]^\top && \text{(D.8)}\\
&= \mathbb{E}\big[\mathbf{x}_i\mathbf{x}_i^\top|\mathbf{W},\sigma^2\big] && \text{(D.9)}\\
&= \int_{\mathbb{R}^d}\mathbf{x}_i\mathbf{x}_i^\top p(\mathbf{x}_i|\mathbf{W},\sigma^2)\,\mathrm{d}\mathbf{x}_i && \text{(D.10)}\\
&= \int_{\mathbb{R}^d}\mathbf{x}_i\mathbf{x}_i^\top\int_{\mathbb{R}^k}p(\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2)p(\mathbf{v}_i)\,\mathrm{d}\mathbf{v}_i\mathrm{d}\mathbf{x}_i && \text{(D.11)}\\
&= \int_{\mathbb{R}^k}\left[\int_{\mathbb{R}^d}\mathbf{x}_i\mathbf{x}_i^\top p(\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2)\,\mathrm{d}\mathbf{x}_i\right]p(\mathbf{v}_i)\mathrm{d}\mathbf{v}_i && \text{(D.12)}\\
&= \mathbb{E}\big[\mathbb{E}\big[\mathbf{x}_i\mathbf{x}_i^\top|\mathbf{v}_i,\mathbf{W},\sigma^2\big]\big] && \text{(D.13)}\\
&= \mathbb{E}\big[\mathbb{E}\big[\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2\big]\mathbb{E}\big[\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2\big]^\top+\operatorname{cov}(\mathbf{x}_i|\mathbf{v}_i,\mathbf{W},\sigma^2)\big] && \text{(D.14)}\\
&= \mathbb{E}\big[\mathbf{W}\mathbf{v}_i\mathbf{v}_i^\top\mathbf{W}^\top+\sigma^2 I\big] && \text{(D.15)}\\
&= \mathbf{W}\mathbb{E}\big[\mathbf{v}_i\mathbf{v}_i^\top\big]\mathbf{W}^\top+\sigma^2 I && \text{(D.16)}\\
&= \mathbf{W}\mathbf{W}^\top+\sigma^2 I && \text{(D.17)}
\end{aligned}$$

Putting all this together gives us

$$p(\mathbf{x}_i|\mathbf{W},\sigma^2)=\mathcal{N}(\mathbf{x}_i|\mathbf{0},\mathbf{W}\mathbf{W}^\top+\sigma^2 I) \tag{D.18}$$

D.2 Approximate Posterior

D.2.1 Structure of $\tilde{\mathbf{H}}_\beta$

Here we provide a detailed derivation of $\tilde{\mathbf{H}}_\beta$ used in the laplace approximation to the evidence in **Section 3.1.2**. The mathematics required is limited only to matrix algebra and matrix derivative operations [Petersen and Pedersen, 2012].

Recall that $\tilde{\mathbf{H}}_\beta$ is block-diagonal, with blocks $\tilde{\mathbf{H}}_\beta^{\mathbf{w}_1}\cdots\tilde{\mathbf{H}}_\beta^{\mathbf{w}_k}$ and $\tilde{\mathbf{H}}_\beta^{\sigma^2}$. Here we derive the form of each of the blocks. This is done by considering just the (l,m)'th element of each block, solving the second partial derivatives, then manipulating the result into a form which allows us to recover a matrix formula for the entire block.

Structure of $\mathbf{H}_\beta^{\mathbf{w}_i}$

The blocks $\tilde{\mathbf{H}}_\beta^{\mathbf{w}_1}\cdots\tilde{\mathbf{H}}_\beta^{\mathbf{w}_k}$ may be derived by considering

$$\frac{\partial^2 -\ln p(\mathbf{X}|\mathbf{W},\sigma^2)p(\mathbf{W}|\beta)}{\partial\mathbf{w}_i\partial\mathbf{w}_j^\top}=\frac{1}{2}\frac{\partial^2}{\partial\mathbf{w}_i\partial\mathbf{w}_j^\top}\left(n\ln|\mathbf{C}|+\operatorname{Tr}\Big[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}^\top\Big]+\operatorname{Tr}\Big[\mathbf{W}^\top\mathbf{K}_\beta^{-1}\mathbf{W}\Big]\right) \tag{D.19}$$

where the first two terms inside the parenthesis come from the likelihood, and the last term comes from the prior. $i, j \in 1 \cdots k$ index the columns of $\mathbf{W}$. We now solve the matrix of partial derivatives of each of these terms separately, then substitute these back in to **Equation D.19**. Note that the second partial derivative is with respect to $\mathbf{w}_i$ and $\mathbf{w}_j$, which may be different. For computing the required blocks only the case of $i = j$ is required. However, looking at all pairs of columns of $\mathbf{W}$ gives us a better insight into how much covariance structure is approximated away when using $\tilde{\mathbf{H}}$ over $\mathbf{H}$.

We start with the impact of the prior.

$$\frac{\partial^2 \operatorname{Tr}\left[\mathbf{W}^\top \mathbf{K}_\beta^{-1} \mathbf{W}\right]}{\partial \mathbf{w}_i \partial \mathbf{w}_j^\top} = \frac{\partial}{\partial \mathbf{w}_j^\top} \sum_{l=1}^{k} \frac{\partial \mathbf{w}_l^\top \mathbf{K}_\beta^{-1} \mathbf{w}_l}{\partial \mathbf{w}_i} \tag{D.20}$$

$$= \frac{\partial}{\partial \mathbf{w}_j^\top} 2\mathbf{K}_\beta^{-1} \mathbf{w}_i \tag{D.21}$$

$$= \mathbb{1}[i = j]\, 2\mathbf{K}_\beta^{-1} \tag{D.22}$$

The indicator function $\mathbb{1}[i = j]$ means that the prior has no impact if $i \neq j$. This is intuitive since the prior is independent on each $\mathbf{w}_i$.

In the following derivation we will be looking at derivatives with respect to the scalars w_{il}, which is the l'th element of the column $\mathbf{w}_i$. We will repeatedly make use of the following matrix-scalar derivative.

$$\frac{\partial \mathbf{C}}{\partial w_{il}} = \frac{\partial \mathbf{W}\mathbf{W}^\top + \sigma^2 I}{\partial w_{il}} = \frac{\partial \mathbf{W}\mathbf{W}^\top}{\partial w_{il}} = \mathbf{W}\mathbf{J}_{il} + \mathbf{J}_{li}\mathbf{W}^\top \tag{D.23}$$

Here, $\mathbf{J}_{il} \in \{0, 1\}^{k \times d}$ is the single-entry matrix; it is zero everywhere except for the (i, l)'th element which is a 1. The properties of the single element matrix are discussed in **Appendix A.2**.

We now consider the first term from the likelihood. Following this, it is useful

to note that the indices i, j run from 1 to k, whilst l, m run from 1 to d.

$$\left(\frac{\partial^2 \ln|\mathbf{C}|}{\partial \mathbf{w}_i \partial \mathbf{w}_j^\top}\right)_{lm} = \frac{\partial}{\partial w_{jm}} \frac{\partial}{\partial w_{il}} \ln|\mathbf{C}| \tag{D.24}$$

Using **Equation A.8**

$$= \frac{\partial}{\partial w_{jm}} \operatorname{Tr}\left[\mathbf{C}^{-1} \frac{\partial \mathbf{C}}{\partial w_{il}}\right] \tag{D.25}$$

Using **Equation D.23**

$$= \frac{\partial}{\partial w_{jm}} \operatorname{Tr}\left[\mathbf{C}^{-1}(\mathbf{W}\mathbf{J}_{il} + \mathbf{J}_{li}\mathbf{W}^\top)\right] \tag{D.26}$$

Split into two sum of two equal traces

$$= 2\frac{\partial}{\partial w_{jm}} \operatorname{Tr}\left[\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\right] \tag{D.27}$$

We can now move the derivative inside the trace (**Equation A.10**) and evaluate the derivative. Both $\mathbf{C}$ and $\mathbf{W}$ depend on w_{jm}, so we require the matrix identities **Equation A.14** and then **Equation A.9**:

$$2\operatorname{Tr}\left[-\mathbf{C}^{-1}\frac{\partial \mathbf{C}}{\partial w_{jm}}\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il} + \mathbf{C}^{-1}\frac{\partial \mathbf{W}}{\partial w_{jm}}\mathbf{J}_{il}\right] \tag{D.28}$$

Expand out matrix derivatives

$$=2\operatorname{Tr}\left[-\mathbf{C}^{-1}(\mathbf{W}\mathbf{J}_{jm} + \mathbf{J}_{mj}\mathbf{W}^\top)\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il} + \mathbf{C}^{-1}\mathbf{J}_{mj}\mathbf{J}_{il}\right] \tag{D.29}$$

Expand

$$=2\operatorname{Tr}\left[\mathbf{C}^{-1}\mathbf{J}_{mj}\mathbf{J}_{il} - \mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{jm}\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il} - \mathbf{C}^{-1}\mathbf{J}_{mj}\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\right] \tag{D.30}$$

This can now split in to three trace terms which are of the form of the single-element matrix identities **Equation A.6** and **Equation A.7**. Using these identities we obtain:

$$2\left(\mathbb{1}[i=j]\left(\mathbf{C}^{-1}\right)_{lm} - \left(\mathbf{C}^{-1}\mathbf{W}\right)_{lj}\left(\mathbf{C}^{-1}\mathbf{W}\right)_{mi} - \left(\mathbf{C}^{-1}\right)_{lm}\left(\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{W}\right)_{ji}\right) \tag{D.31}$$

$$=2\left(\mathbb{1}[i=j]\left(\mathbf{C}^{-1}\right)_{lm} - \left(\mathbf{C}^{-1}\mathbf{w}_j\right)_l\left(\mathbf{C}^{-1}\mathbf{w}_i\right)_m - \mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i\left(\mathbf{C}^{-1}\right)_{lm}\right) \tag{D.32}$$

Where the second line has been obtained using $\left(\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{W}\right)_{ji} = \mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i$. We now use $\left(\mathbf{C}^{-1}\mathbf{w}_j\right)_l\left(\mathbf{C}^{-1}\mathbf{w}_i\right)_m = \left(\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1}\right)_{lm}$ and move the subscripting to

outer parentheses:

$$2\Big(\mathbb{1}[i=j]\,\mathbf{C}^{-1}-\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1}-\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i\mathbf{C}^{-1}\Big)_{lm} \tag{D.33}$$

Since this is true for all $l, m \in 1\cdots d$ we can make make the statement

$$\frac{\partial^2 \ln|\mathbf{C}|}{\partial\mathbf{w}_i\partial\mathbf{w}_j^\top}=2\Big(\mathbb{1}[i=j]\,\mathbf{C}^{-1}-\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1}-\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i\mathbf{C}^{-1}\Big) \tag{D.34}$$

We now consider the $\mathrm{Tr}\big[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}^\top\big]$ term. First we move the derivative with respect to w_{il} inside the trace, use the identity for the derivative of the inverse of a matrix, then evaluate $\frac{\partial\mathbf{C}}{\partial w_{il}}$:

$$\left(\frac{\partial^2\,\mathrm{Tr}\big[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}^\top\big]}{\partial\mathbf{w}_i\partial\mathbf{w}_j^\top}\right)_{lm}=-\frac{\partial}{\partial w_{jm}}\frac{\partial}{\partial w_{il}}\,\mathrm{Tr}\Big[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}^\top\Big] \tag{D.35}$$

$$=-\frac{\partial}{\partial w_{jm}}\,\mathrm{Tr}\left[\mathbf{X}\frac{\partial\mathbf{C}^{-1}}{\partial w_{il}}\mathbf{X}^\top\right] \tag{D.36}$$

$$=-\frac{\partial}{\partial w_{jm}}\,\mathrm{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial w_{il}}\mathbf{C}^{-1}\mathbf{X}^\top\right] \tag{D.37}$$

$$=-\frac{\partial}{\partial w_{jm}}\,\mathrm{Tr}\Big[\mathbf{X}\mathbf{C}^{-1}(\mathbf{W}\mathbf{J}_{il}+\mathbf{J}_{li}\mathbf{W}^\top)\mathbf{C}^{-1}\mathbf{X}^\top\Big] \tag{D.38}$$

$$=-2\frac{\partial}{\partial w_{jm}}\,\mathrm{Tr}\Big[\mathbf{X}\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1}\mathbf{X}^\top\Big] \tag{D.39}$$

We now move $\frac{\partial}{\partial w_{jm}}$ inside the trace and use the identity for taking the deriv-

ative of a product of matrices. We then expand out the parentheses and rearrange:

$$-2\,\mathrm{Tr}\left[\mathbf{X}^\top\mathbf{X}\frac{\partial\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1}}{\partial w_{jm}}\right] \tag{D.40}$$

$$\begin{aligned}
&= -2\,\mathrm{Tr}\Big[\mathbf{X}^\top\mathbf{X}\Big(-\mathbf{C}^{-1}(\mathbf{W}\mathbf{J}_{jm}+\mathbf{J}_{mj}\mathbf{W}^\top)\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1} \\
&\qquad\qquad + \mathbf{C}^{-1}\mathbf{J}_{mj}\mathbf{J}_{il}\mathbf{C}^{-1} \\
&\qquad\qquad -\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1}(\mathbf{W}\mathbf{J}_{jm}+\mathbf{J}_{mj}\mathbf{W}^\top)\mathbf{C}^{-1}\Big)\Big]
\end{aligned} \tag{D.41}$$

$$\begin{aligned}
&= 2\,\mathrm{Tr}\Big[\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\Big(\mathbf{W}\mathbf{J}_{jm}\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il} \quad + \\
&\qquad\qquad \mathbf{J}_{mj}\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{il} \quad + \\
&\qquad\qquad \mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1}\mathbf{W}\mathbf{J}_{jm} \quad + \\
&\qquad\qquad \mathbf{W}\mathbf{J}_{il}\mathbf{C}^{-1}\mathbf{J}_{mj}\mathbf{W}^\top \quad - \\
&\qquad\qquad \mathbf{J}_{mj}\mathbf{J}_{il}\Big)\Big]
\end{aligned} \tag{D.42}$$

Finally we use identities of the single-element matrix inside of a trace. This removes the trace operator and all single-element matrices. After this we use the fact that all i and j subscripts pull out a single column of $\mathbf{W}$, so we can instead refer to $\mathbf{w}_i$ and $\mathbf{w}_j$. We can then move the l, m subscripts to the outer parentheses:

$$\begin{aligned}
&= 2\Big((\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{W})_{lj}(\mathbf{C}^{-1}\mathbf{W})_{mi} \quad + \\
&\qquad (\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1})_{lm}(\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{W})_{ji} \quad + \\
&\qquad (\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{W})_{mi}(\mathbf{C}^{-1}\mathbf{W})_{lj} \quad + \\
&\qquad (\mathbf{W}^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{W})_{ji}(\mathbf{C}^{-1})_{lm} \quad - \\
&\qquad (\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1})_{lm}\mathbb{1}[i=j]\Big)
\end{aligned} \tag{D.43}$$

$$\begin{aligned}
&= 2\Big(\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1} \quad + \\
&\qquad \mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} \quad + \\
&\qquad (\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} \quad + \\
&\qquad (\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1} \quad - \\
&\qquad \mathbb{1}[i=j]\,\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\Big)_{lm}
\end{aligned} \tag{D.44}$$

$$\tag{D.45}$$

Again, since this is true for all indices l, m we can state the entire matrix at once

$$\begin{aligned}\frac{\partial^2 \operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}^\top\right]}{\partial \mathbf{w}_i \partial \mathbf{w}_j^\top} = 2\Big(&\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1} &+\\ &\mathbf{C}^{-1}\mathbf{w}_j\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} &+\\ &(\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} &+\\ &(\mathbf{w}_j^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1} &-\\ &\mathbb{1}[i=j]\,\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\Big)\end{aligned} \tag{D.46}$$

Putting these together we arrive at the definition of each of the blocks:

$$\begin{aligned}\mathbf{H}_\beta^{\mathbf{w}_i} = \Big(&n\mathbf{C}^{-1} &-\\ &n\mathbf{C}^{-1}\mathbf{w}_i\mathbf{w}_i^\top\mathbf{C}^{-1} &-\\ &n\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{w}_i\mathbf{C}^{-1} &+\\ &\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i\mathbf{w}_i^\top\mathbf{C}^{-1} &+\\ &\mathbf{C}^{-1}\mathbf{w}_i\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} &+\\ &(\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} &+\\ &(\mathbf{w}_i^\top\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1}\mathbf{w}_i)\mathbf{C}^{-1} &-\\ &\mathbf{C}^{-1}\mathbf{X}^\top\mathbf{X}\mathbf{C}^{-1} &+\\ &\mathbf{K}_\beta^{-1}\Big)\end{aligned} \tag{D.47}$$

Structure of $\mathbf{H}_\beta^{\sigma^2}$

$$\mathbf{H}_\beta^{\sigma^2} = \frac{\partial^2 -\ln p(\mathbf{X}|\mathbf{W},\sigma^2)p(\sigma^2)}{(\partial\sigma^2)^2} \tag{D.48}$$

$$= \frac{1}{2}\frac{\partial^2}{(\partial\sigma^2)^2}\left(n\ln|\mathbf{C}| + \operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}\right] - 2\ln\sigma^{-2}\right) \tag{D.49}$$

We derive the three terms separately below, then put them all together.

$$\frac{\partial^2 \ln|\mathbf{C}|}{(\partial\sigma^2)^2} = \frac{\partial}{\partial\sigma^2}\operatorname{Tr}\left[\mathbf{C}^{-1}\frac{\partial\mathbf{C}}{\partial\sigma^2}\right] \tag{D.50}$$

$$= \frac{\partial}{\partial\sigma^2}\operatorname{Tr}\left[\mathbf{C}^{-1}\right] \tag{D.51}$$

$$= -\operatorname{Tr}\left[\mathbf{C}^{-1}\mathbf{C}^{-1}\right] \tag{D.52}$$

$$\tag{D.53}$$

$$\frac{\partial^2 \operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{X}\right]}{(\partial\sigma^2)^2} = -\frac{\partial}{\partial\sigma^2}\operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{X}^\top\right] \tag{D.54}$$

$$= -\operatorname{Tr}\left[\mathbf{X}\frac{\partial\mathbf{C}^{-1}\mathbf{C}^{-1}}{\partial\sigma^2}\mathbf{X}^\top\right] \tag{D.55}$$

$$= 2\operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{X}^\top\right] \tag{D.56}$$

$$\frac{\partial^2 \ln\sigma^{-2}}{(\partial\sigma^2)^2} = \sigma^{-4} \tag{D.57}$$

Using these we can obtain an equation for $\mathbf{H}_\beta^{\sigma^2}$:

$$\mathbf{H}_\beta^{\sigma^2} = \operatorname{Tr}\left[\mathbf{X}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{C}^{-1}\mathbf{X}^\top\right] - \frac{n}{2}\left[\mathbf{C}^{-1}\mathbf{C}^{-1}\right] - \sigma^{-4} \tag{D.58}$$

Bibliography

Ramesh P. Arasaradnam, Nathalie Ouaret, Matthew G. Thomas, Nabil Quraishi, Evelyn Heatherington, Chuka U. Nwokolo, Karna D. Bardhan, and James A. Covington. A novel tool for noninvasive diagnosis and tracking of patients with inflammatory bowel disease. *Inflammatory Bowel Diseases*, 19(5):999–1003, 2013. URL `http://dx.doi.org/10.1097/MIB.0b013e3182802b26`.

Ramesh P. Arasaradnam, Michael J. McFarlane, Courtenay Ryan-Fisher, Erik Westenbrink, Paula Hodges, Matthew G. Thomas, Samantha Chambers, Nicola O'Connell, Catherine Bailey, Christopher Harmston, Chuka U. Nwokolo, Karna D. Bardhan, and James A. Covington. Detection of colorectal cancer (crc) by urinary volatile organic compound analysis. *PLOS ONE*, 9(9):1–6, 09 2014. doi: 10.1371/journal.pone.0108750. URL `https://doi.org/10.1371/journal.pone.0108750`.

Ramesh P Arasaradnam, Michael McFarlane, Emma Daulton, Erik Westenbrink, Nicola O'Connell, Subiatu Wurie, Chuka U Nwokolo, Karna D Bardhan, Richard S Savage, and James A Covington. Non-invasive distinction of non-alcoholic fatty liver disease using urinary volatile organic compound analysis: early results. *Journal of Gastrointestinal & Liver Diseases*, 24(2), 2015.

Ramesh P Arasaradnam, M McFarlane, K Ling, S Wurie, N O'Connell, Chuka U Nwokolo, Karna Dev Bardhan, J Skinner, Richard S Savage, and James A Covington. Breathomicsexhaled volatile organic compound analysis to detect hepatic encephalopathy: a pilot study. *Journal of breath research*, 10(1):1–6, 2016a. ISSN 1752-7163.

Ramesh P. Arasaradnam, Michael McFarlane, Emma Daulton, Jim Skinner, Nicola O'Connell, Subiatu Wurie, Samantha Chambers, Chuka Nwokolo, Karna Bardhan, Richard Savage, and James Covington. Non-invasive exhaled volatile organic biomarker analysis to detect inflammatory bowel disease (IBD). *Digestive and Liver Disease*, 48(2):148 – 153, 2016b. ISSN 1590-8658. doi:

http://dx.doi.org/10.1016/j.dld.2015.10.013. URL http://www.sciencedirect.com/science/article/pii/S1590865815301067.

Pierre Baldi and Kurt Hornik. Neural networks and principal component analysis: Learning from examples without local minima. *Neural Networks*, 2(1):53 – 58, 1989. ISSN 0893-6080. doi: https://doi.org/10.1016/0893-6080(89)90014-2. URL http://www.sciencedirect.com/science/article/pii/0893608089900142.

Daniel J. C. Berkhout, Marc A. Benninga, Ruby M. van Stein, Paul Brinkman, Hendrik J. Niemarkt, Nanne K. H. de Boer, and Tim G. J. de Meij. Effects of Sampling Conditions and Environmental Factors on Fecal Volatile Organic Compound Analysis by an Electronic Nose Device. *Sensors*, 16, 2016. doi: 10.3390/s16111967. URL http://www.mdpi.com/1424-8220/16/11/1967.

M. Bernabei, G. Pennazza, M. Santonico, C. Corsi, C. Roscioni, R. Paolesse, C. Di Natale, and A. DAmico. A preliminary study on the possibility to diagnose urinary tract cancers by an electronic nose. *Sensors and Actuators B: Chemical*, 131, 2008. ISSN 0925-4005. doi: https://doi.org/10.1016/j.snb.2007.12.030. URL http://www.sciencedirect.com/science/article/pii/S0925400507009872.

Rajendra Bhatia and Peter Rosenthal. How and Why to Solve the Operator Equation AXXB = Y. *Bulletin of the London Mathematical Society*, 29(1):1–21, 1997. ISSN 1469-2120. doi: 10.1112/S0024609396001828. URL http://dx.doi.org/10.1112/S0024609396001828.

Christopher M. Bishop. *Pattern Recognition and Machine Learning*. Springer, 2006.

ke Björck and Gene H Golub. Numerical methods for computing angles between linear subspaces. *Mathematics of computation*, 27(123):579–594, 1973.

L Blanchet, A Smolinska, A Baranska, E Tigchelaar, M Swertz, A Zhernakova, J W Dallinga, C Wijmenga, and F J van Schooten. Factors that influence the volatile organic compound content in human breath. *Journal of Breath Research*, 11(1):016013, 2017. URL http://stacks.iop.org/1752-7163/11/i=1/a=016013.

Marije K Bomers, Frederik P Menke, Richard S Savage, Christina MJE Vandenbroucke-Grauls, Michiel A Van Agtmael, James A Covington, and Yvo M Smulders. Rapid, accurate, and on-site detection of c. difficile in stool samples. *The American journal of gastroenterology*, 110(4):588–594, 2015.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, Oct 2001. ISSN 1573-0565.

P. Brinkman, M. A. Pol, M. G. Gerritsen, L. D. Bos, T. Dekker, B. S. Smids, A. Sinha, C. J. Majoor, M. M. Sneeboer, H. H. Knobel, T. J. Vink, F. H. Jongh, R. Lutter, P. J. Sterk, and N. Fens. Exhaled breath profiles in the monitoring of loss of control and clinical recovery in asthma. *Clinical & Experimental Allergy*, 2017. doi: 10.1111/cea.12965. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cea.12965.

Marcel Bruins, Zeaur Rahim, Albert Bos, Wendy W. J. van de Sande, Hubert Ph Endtz, and Alex van Belkum. Diagnosis of active tuberculosis by e-nose analysis of exhaled air. *Tuberculosis*, pages 232–238, 2013. ISSN 1472-9792.

Emmanuel J. Candès, Xiaodong Li, Yi Ma, and John Wright. Robust principal component analysis? *J. ACM*, 58(3):11:1–11:37, June 2011. ISSN 0004-5411. URL http://doi.acm.org/10.1145/1970392.1970395.

L. Carmel, S. Levy, D. Lancet, and D. Harel. A feature extraction method for chemical sensors in electronic noses. *Sensors and Actuators B: Chemical*, 93(1):67 – 76, 2003. ISSN 0925-4005. doi: http://dx.doi.org/10.1016/S0925-4005(03)00247-8. URL http://www.sciencedirect.com/science/article/pii/S0925400503002478. Proceedings of the Ninth International Meeting on Chemical Sensors.

Bob Carpenter, Andrew Gelman, Matthew Hoffman, Daniel Lee, Ben Goodrich, Michael Betancourt, Marcus Brubaker, Jiqiang Guo, Peter Li, and Allen Riddell. Stan: A probabilistic programming language. *Journal of Statistical Software, Articles*, 76(1):1–32, 2017. ISSN 1548-7660. doi: 10.18637/jss.v076.i01. URL https://www.jstatsoft.org/v076/i01.

W.J. Cash, P. McConville, E. McDermott, P.A. McCormick, M.E. Callender, and N.I. McDougall. Current concepts in the assessment and treatment of hepatic encephalopathy. *QJM: An International Journal of Medicine*, 103(1):9, 2010. doi: 10.1093/qjmed/hcp152. URL +http://dx.doi.org/10.1093/qjmed/hcp152.

F.K. Che Harun, J.A. Covington, and J.W. Gardner. Mimicking the biological olfactory system: a portable electronic mucosa. *IET Nanobiotechnology*, 2012. URL http://digital-library.theiet.org/content/journals/10.1049/iet-nbt.2010.0032.

J. A. Covington, M. P. van. der Schee, A. S. L. Edge, B. Boyle, R. S. Savage, and R. P. Arasaradnam. The application of faims gas analysis in medical diagnostics.

Analyst, 140:6775–6781, 2015. doi: 10.1039/C5AN00868A. URL http://dx.doi.org/10.1039/C5AN00868A.

Morris H. DeGroot and Mark J. Schervish. *Probability and Statistics*. Pearson Education Limited, 2013. ISBN 9781292037677.

Silvano Dragonieri, Robert Schot, Bart J.A. Mertens, Saskia Le Cessie, Stefanie A. Gauw, Antonio Spanevello, Onofrio Resta, Nico P. Willard, Teunis J. Vink, Klaus F. Rabe, Elisabeth H. Bel, and Peter J. Sterk. An electronic nose in the discrimination ofpatients with asthma and controls. *Journal of Allergy and Clinical Immunology*, 120(4):856 – 862, 2007. ISSN 0091-6749. doi: http://dx.doi.org/10.1016/j.jaci.2007.05.043. URL http://www.sciencedirect.com/science/article/pii/S009167490701038X.

Silvano Dragonieri, Jouke T. Annema, Robert Schot, Marc P.C. van der Schee, Antonio Spanevello, Pierluigi Carrat, Onofrio Resta, Klaus F. Rabe, and Peter J. Sterk. An electronic nose in the discrimination of patients with non-small cell lung cancer and copd. *Lung Cancer*, 64(2):166 – 170, 2009. ISSN 0169-5002. doi: http://dx.doi.org/10.1016/j.lungcan.2008.08.008. URL http://www.sciencedirect.com/science/article/pii/S0169500208004194.

Arnaldo DAmico, Giorgio Pennazza, Marco Santonico, Eugenio Martinelli, Claudio Roscioni, Giovanni Galluccio, Roberto Paolesse, and Corrado Di Natale. An investigation on electronic nose diagnosis of lung cancer. *Lung Cancer*, 68(2):170 – 176, 2010. ISSN 0169-5002. doi: http://dx.doi.org/10.1016/j.lungcan.2009.11.003. URL http://www.sciencedirect.com/science/article/pii/S0169500209005807.

Gary Alan Eiceman, Zeev Karpas, and Herbert H Hill Jr. *Ion mobility spectrometry*. CRC press, 2013.

S Esfahani, J. R. Skinner, R. S. Savage, N O'Connell, I Kyrou, C. U. Nwokolo, K. D. Bardhan, R. P. Arasaradnam, and J. A. Covington. A rapid discrimination of diabetic patients from volunteers using urinary volatile and an electronic nose. June 2015.

Siavash Esfahani, Nidhi M. Sagar, Ioannis Kyrou, Ella Mozdiak, Nicola O'Connell, Chuka Nwokolo, Karna D. Bardhan, Ramesh P. Arasaradnam, and James A. Covington. Variation in gas and volatile compound emissions from human urine as it ages, measured by an electronic nose. *Biosensors*, 6, 2016. ISSN 2079-6374. doi: 10.3390/bios6010004. URL http://www.mdpi.com/2079-6374/6/1/4.

N. Fens, M. P. Schee, P. Brinkman, and P. J. Sterk. Exhaled breath analysis by electronic nose in airways disease. established issues and key questions. *Clinical & Experimental Allergy*, 43(7):705–715, 2012. doi: 10.1111/cea.12052. URL https://onlinelibrary.wiley.com/doi/abs/10.1111/cea.12052.

L. Fernandez, A. Martn-Gmez, M. Mar Contreras, M. Padilla, S. Marco, and L. Arce. Ham quality evaluation assisted by gas chromatography ion mobility spectrometry. In *2017 ISOCS/IEEE International Symposium on Olfaction and Electronic Nose (ISOEN)*, pages 1–3, May 2017. doi: 10.1109/ISOEN.2017.7968852.

Shai Fine and Katya Scheinberg. Efficient svm training using low-rank kernel representations. *Journal of Machine Learning Research*, 2(Dec):243–264, 2001.

R. Fletcher. *Practical Methods of Optimization; (2Nd Ed.)*. Wiley-Interscience, New York, NY, USA, 1987. ISBN 0-471-91547-5.

I.A. Fowlis. *Gas chromatography: analytical chemistry by open learning*. Analytical Chemistry by Open Learning. Wiley, 1995. ISBN 9780471954682. URL https://books.google.co.uk/books?id=fzcvAQAAIAAJ.

Jerome Friedman, Trevor Hastie, and Robert Tibshirani. Regularization paths for generalized linear models via coordinate descent. *Journal of Statistical Software*, 33(1):1–22, 2010. URL http://www.jstatsoft.org/v33/i01/.

J W Gardner and PN Bartlett. Electronic noses. principles and applications. *Measurement Science and Technology*, 11(7):1087, 2000. URL http://stacks.iop.org/0957-0233/11/i=7/a=702.

Rocío Garrido-Delgado, Lourdes Arce, and Miguel Valcárcel. Multi-capillary column-ion mobility spectrometry: a potential screening system to differentiate virgin olive oils. *Analytical and Bioanalytical Chemistry*, 402(1):489–498, Jan 2012. ISSN 1618-2650. doi: 10.1007/s00216-011-5328-1. URL https://doi.org/10.1007/s00216-011-5328-1.

FlavourSpec with PAL RSI User Manual. G.A.S. Gesellschaft für analytische Sensorsysteme mbH, Otto-Hahn-Straße 15, 4227 Dortmund, Germany, 1.0.1 edition, Feb 2017.

A. Gelman, J.B. Carlin, H.S. Stern, D.B. Dunson, A. Vehtari, and D.B. Rubin. *Bayesian Data Analysis, Third Edition*. Chapman & Hall/CRC Texts in Statistical Science. Taylor & Francis, 2013. ISBN 9781439840955. URL https://books.google.co.uk/books?id=ZXL6AQAAQBAJ.

Yoav Gilad and Doron Lancet. Population differences in the human functional olfactory repertoire. *Molecular Biology and Evolution*, 20(3):307–314, 2003. doi: 10.1093/molbev/msg013. URL http://dx.doi.org/10.1093/molbev/msg013.

Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. http://www.deeplearningbook.org.

Yue Guan and Jennifer Dy. Sparse probabilistic principal component analysis. In David van Dyk and Max Welling, editors, *Proceedings of the Twelth International Conference on Artificial Intelligence and Statistics*, volume 5, pages 185–192. PMLR, 2009. URL http://proceedings.mlr.press/v5/guan09a.html.

Roger Guevremont. High-field asymmetric waveform ion mobility spectrometry: a new tool for mass spectrometry. *Journal of Chromatography A*, 1058(1):3–19, 2004.

Trevor Hastie, Robert Tibshirani, and Martin Wainwright. *Statistical Learning with Sparsity: The Lasso and Generalizations*. Chapman & Hall/CRC, 2015. ISBN 1498712169, 9781498712163.

Myles Hollander, Douglas A Wolfe, and Eric Chicken. *Nonparametric statistical methods*. John Wiley & Sons, 2013.

Mia Hubert, Tom Reynkens, Eric Schmitt, and Tim Verdonck. Sparse pca for high-dimensional data with outliers. *Technometrics*, 58(4):424–434, 2016. doi: 10.1080/00401706.2015.1093962. URL https://doi.org/10.1080/00401706.2015.1093962.

A. Hyvrinen and E. Oja. Independent component analysis: algorithms and applications. *Neural Networks*, 13(4):411 – 430, 2000. ISSN 0893-6080. doi: https://doi.org/10.1016/S0893-6080(00)00026-5. URL http://www.sciencedirect.com/science/article/pii/S0893608000000265.

John PA Ioannidis. Why most published research findings are false. *PLoS medicine*, 2(8):e124, 2005.

David James, Simon M. Scott, Zulfiqur Ali, and William T. O'Hare. Chemical sensors for electronic nose systems. *Microchimica Acta*, 149(1):1–17, Feb 2005. ISSN 1436-5073. doi: 10.1007/s00604-004-0291-6. URL http://dx.doi.org/10.1007/s00604-004-0291-6.

E. T. Jaynes. *Probability theory: The logic of science*. Cambridge University Press, Cambridge, 2003.

Harold Jeffreys. *The Theory of Probability*. Oxford, third edition, 1998.

MultiSens Analyser Manual. JLM Innovation, 08 2009. Version 0.99.

Ian T Jolliffe, Nickolay T Trendafilov, and Mudassir Uddin. A modified principal component technique based on the lasso. *Journal of Computational and Graphical Statistics*, 12(3):531–547, 2003. doi: 10.1198/1061860032148. URL `https://doi.org/10.1198/1061860032148`.

Alexandros Karatzoglou, Alex Smola, Kurt Hornik, and Achim Zeileis. kernlab – an S4 package for kernel methods in R. *Journal of Statistical Software*, 11(9):1–20, 2004. URL `http://www.jstatsoft.org/v11/i09/`.

Robert E. Kass and Adrian E. Raftery. Bayes factors. *Journal of the American Statistical Association*, 90(430):773–795, 1995. doi: 10.1080/01621459.1995.10476572. URL `http://amstat.tandfonline.com/doi/abs/10.1080/01621459.1995.10476572`.

Mohammad E Khan, Guillaume Bouchard, Kevin P Murphy, and Benjamin M Marlin. Variational bounds for mixed-data factor analysis. In J. D. Lafferty, C. K. I. Williams, J. Shawe-Taylor, R. S. Zemel, and A. Culotta, editors, *Advances in Neural Information Processing Systems 23*, pages 1108–1116. Curran Associates, Inc., 2010. URL `http://papers.nips.cc/paper/3947-variational-bounds-for-mixed-data-factor-analysis.pdf`.

Andrew V Knyazev and Peizhen Zhu. Principal angles between subspaces and their tangents. *Arxiv preprint*, 2012.

A. H. J. Kolk, J. J. B. N. van Berkel, M. M. Claassens, E. Walters, S. Kuijper, J. W. Dallinga, and F. J. van Schooten. Breath analysis as a potential diagnostic tool for tuberculosis. *The International Journal of Tuberculosis and Lung Disease*, 16 (6):777–782, 2012.

Arend Kolk, Michael Hoelscher, Leonard Maboko, Jutta Jung, Sjoukje Kuijper, Michael Cauchi, Conrad Bessant, Stella van Beers, Ritaban Dutta, Tim Gibson, and Klaus Reither. Electronic-nose technology using sputum samples in diagnosis of patients with tuberculosis. *Journal of clinical microbiology*, 48(11):4235–4238, 2010.

E. V. Krisilova, A. M. Levina, and V. A. Makarenko. Determination of the volatile compounds of vegetable oils using an ion-mobility spectrometer. *Journal of*

Analytical Chemistry, 69(4):371–376, Apr 2014. ISSN 1608-3199. doi: 10.1134/S1061934814020075. URL https://doi.org/10.1134/S1061934814020075.

Max Kuhn and Kjell Johnson. *Applied predictive modeling*, volume 26. Springer, 2013.

Neil D Lawrence. Gaussian process latent variable models for visualisation of high dimensional data. In *Advances in Neural Information Processing Systems*, pages 329–336, 2004.

Joseph M. Lewis, Richard S. Savage, Nicholas J. Beeching, Mike B. J. Beadsworth, Nicholas Feasey, and James A. Covington. Identifying volatile metabolite signatures for the diagnosis of bacterial respiratory tract infection using electronic nose technology: A pilot study. *PLOS ONE*, 2017. URL https://doi.org/10.1371/journal.pone.0188879.

Andy Liaw and Matthew Wiener. Classification and regression by randomforest. *R News*, 2(3):18–22, 2002. URL http://CRAN.R-project.org/doc/Rnews/.

Z. Lin, M. Chen, and Y. Ma. The Augmented Lagrange Multiplier Method for Exact Recovery of Corrupted Low-Rank Matrices. *ArXiv e-prints*, September 2010.

James Robert Lloyd, David Duvenaud, Roger Grosse, Joshua B. Tenenbaum, and Zoubin Ghahramani. Automatic construction and Natural-Language description of nonparametric regression models. In *Association for the Advancement of Artificial Intelligence (AAAI)*, 2014.

David J. C. MacKay. *Information Theory, Inference & Learning Algorithms*. Cambridge University Press, New York, NY, USA, 2002. ISBN 0521642981.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online dictionary learning for sparse coding. In *Proceedings of the 26th Annual International Conference on Machine Learning*, ICML '09, pages 689–696, New York, NY, USA, 2009. ACM. ISBN 978-1-60558-516-1. doi: 10.1145/1553374.1553463. URL http://doi.acm.org/10.1145/1553374.1553463.

Julien Mairal, Francis Bach, Jean Ponce, and Guillermo Sapiro. Online learning for matrix factorization and sparse coding. *J. Mach. Learn. Res.*, 11:19–60, March 2010. ISSN 1532-4435. URL http://dl.acm.org/citation.cfm?id=1756006.1756008.

Isabel Márquez-Sillero, Soledad Crdenas, Stefanie Sielemann, and Miguel Valcrcel. On-line headspace-multicapillary column-ion mobility spectrometry hyphenation as a tool for the determination of off-flavours in foods. *Journal of Chromatography A*, 1333:99 – 105, 2014. ISSN 0021-9673. doi: https://doi.org/10.1016/j.chroma.2014.01.062. URL `http://www.sciencedirect.com/science/article/pii/S0021967314001447`.

Andrea S. Martinez-Vernon, James A. Covington, Ramesh P. Arasaradnam, Siavash Esfahani, Nicola O'Connell, Ioannis Kyrou, and Richard S. Savage. An improved machine learning pipeline for urinary volatiles disease detection: Diagnosing diabetes. Unpublished article. Author webpage at `https://warwick.ac.uk/study/csde/gsp/eportfolio/directory/pg/live/smrnab/`.

Arman Melkumyan and Fabio Ramos. A sparse covariance function for exact gaussian process inference in large datasets. In *IJCAI*, volume 9, pages 1936–1942, 2009.

Paolo Montuschi, Marco Santonico, Chiara Mondino, Giorgio Pennazza, Giulia Mantini, Eugenio Martinelli, Rosamaria Capuano, Giovanni Ciabattoni, Roberto Paolesse, Corrado Di Natale, et al. Diagnostic performance of an electronic nose, fractional exhaled nitric oxide, and lung function testing in asthma. *CHEST Journal*, 137(4):790–796, 2010.

Paolo Montuschi, Nadia Mores, Andrea Trové, Chiara Mondino, and Peter J Barnes. The electronic nose in respiratory medicine. *Respiration*, 85(1):72–84, 2013.

Emily Moser and Michael McCulloch. Canine scent detection of human cancers: A review of methods and accuracy. *Journal of Veterinary Behavior: Clinical Applications and Research*, 5(3):145 – 152, 2010. URL `http://www.sciencedirect.com/science/article/pii/S1558787810000031`.

Kevin P. Murphy. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012. ISBN 9780262018029.

John C. Nash and Ravi Varadhan. Unifying optimization algorithms to aid software system users: optimx for R. *Journal of Statistical Software*, 43(9):1–14, 2011. URL `http://www.jstatsoft.org/v43/i09/`.

Guy Nason. *wavethresh: Wavelets Statistics and Transforms*, 2016. URL `http://CRAN.R-project.org/package=wavethresh`. R package version 4.6.8.

National Guideline Clearinghouse. Uk national guideline for the management of bacterial vaginosis 2012. *National Guideline Clearinghouse*, 2012. URL https://www.guideline.gov/summaries/summary/37218.

National Institute for Health and Care Excellence. Crohn's disease: management. *NICE guidelines*, October 2012. URL https://www.nice.org.uk/guidance/cg152.

National Institute for Health and Care Excellence. Ulcerative colitis: management. *NICE guidelines*, June 2013. URL https://www.nice.org.uk/guidance/cg166.

National Institute for Health and Care Excellence. Inflammatory bowel disease. *NICE guidelines*, February 2015. URL https://www.nice.org.uk/guidance/cg166.

Bruno A. Olshausen and David J. Field. Sparse coding with an overcomplete basis set: A strategy employed by v1? *Vision Research*, 37(23):3311 – 3325, 1997. ISSN 0042-6989. doi: https://doi.org/10.1016/S0042-6989(97)00169-7. URL http://www.sciencedirect.com/science/article/pii/S0042698997001697.

Owlstone Nanotech. Owlstone nanotech white paper. Technical Report OWL-WP-1 v3.0, St Johns Innovation Centre, Cowley Road, Cambridge, CB4 0WS, 2006. URL http://info.owlstonenanotech.com/rs/owlstone/images/FAIMS%20Whitepaper.pdf.

Owlstone Nanotech. *Lonestar Datasheet*. Owlstone Nanotech, 127 Cambridge Science Park, Milton Road, Cambridge, CB4 0GD, June 2017a. URL http://info.owlstonenanotech.com/rs/owlstone/images/Lonestar_2Pager.pdf.

Owlstone Nanotech. *Lonestar user manual, quick start up, maintenance and troubleshooting guides*. Owlstone Nanotech, 127 Cambridge Science Park, Milton Road, Cambridge, CB4 0GD, June 2017b. URL http://support.owlstonenanotech.com/hc/en-us/categories/201660806-Lonestar-and-FAIMS-CORE-PAD-.

Pentti Paatero and Unto Tapper. Positive matrix factorization: A non-negative factor model with optimal utilization of error estimates of data values. *Environmetrics*, 5(2):111–126, 1994. ISSN 1099-095X. doi: 10.1002/env.3170050203. URL http://dx.doi.org/10.1002/env.3170050203.

Trevor Park and George Casella. The bayesian lasso. *Journal of the American Statistical Association*, 103(482):681–686, 2008. doi: 10.1198/016214508000000337. URL https://doi.org/10.1198/016214508000000337.

Krishna Persaud and George Dodd. Analysis of discrimination mechanisms in the mammalian olfactory system using a model nose. *Nature*, 299(5881):352–355, 1982.

K. B. Petersen and M. S. Pedersen. The matrix cookbook, November 2012. URL http://www2.imm.dtu.dk/pubdb/p.php?3274.

R Core Team. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria, 2014. URL http://www.R-project.org/.

Glen C Rains, Samuel L Utley, and W Joe Lewis. Behavioral monitoring of trained insects for chemical detection. *Biotechnology progress*, 22(1):2–8, 2006.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian Processes for Machine Learning*. The MIT Press, 2005. ISBN 026218253X.

Christian P. Robert. *The Bayesian Choice: From Decision-Theoretic Foundations to Computational Implementation*. Springer, 2nd edition, May 2007. ISBN 978-0-387-71598-8.

Donald B. Rubin and Dorothy T. Thayer. Em algorithms for ml factor analysis. *Psychometrika*, 47(1):69–76, Mar 1982. ISSN 1860-0980. doi: 10.1007/BF02293851. URL https://doi.org/10.1007/BF02293851.

Nidhi M. Sagar, Ian A. Cree, James A. Covington, and Ramesh P. Arasaradnam. The interplay of the gut microbiome, bile acids, and volatile organic compounds. *Gastroenterology Research and Practice*, 2015.

Amandip S Sahota, Ravi Gowda, Ramesh P Arasaradnam, Emma Daulton, Richard S Savage, Jim R Skinner, Emily Adams, Stephen A Ward, and James A Covington. A simple breath test for tuberculosis using ion mobility: A pilot study. *Tuberculosis*, 99:143–146, 2016.

Daniel Schacter, Daniel Gilbert, Daniel Wegner, and Matthew Nock. *Psychology*. Macmillan Learning, 2014. ISBN 1464106037.

Alexandre A. Shvartsburg, Richard D. Smith, Ashley Wilks, Andrew Koehl, David Ruiz-Alonso, and Billy Boyle. Ultrafast differential ion mobility spectrometry at

extreme electric fields in multichannel microchips. *Analytical Chemistry*, 81(15): 6489–6495, 2009. doi: 10.1021/ac900892u. URL http://dx.doi.org/10.1021/ac900892u. PMID: 19583243.

Joseph P. Simmons, Leif D. Nelson, and Uri Simonsohn. False-positive psychology. *Psychological Science*, 22(11):1359–1366, 2011. doi: 10.1177/0956797611417632. URL http://dx.doi.org/10.1177/0956797611417632. PMID: 22006061.

R. Tibshirani. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society (Series B)*, pages 267–288, 1996.

Michael E. Tipping and Chris M. Bishop. Probabilistic principal component analysis. *Journal of the Royal Statistical Society, Series B*, 61:611–622, 1999.

Lloyd N. Trefethen and David Bau. *Numerical Linear Algebra*. SIAM, 1997. ISBN 0898713617.

Nora van Gaal, Rozanne Lakenman, James Covington, Richard Savage, Evelien de Groot, Marije Bomers, Marc Benninga, Chris Mulder, Nanne de Boer, and Tim de Meij. Faecal volatile organic compounds analysis using field asymmetric ion mobility spectrometry: non-invasive diagnostics in paediatric inflammatory bowel disease. *Journal of Breath Research*, 2017. URL http://iopscience.iop.org/10.1088/1752-7163/aa6f1d.

Holger Wendland. Piecewise polynomial, positive definite and compactly supported radial functions of minimal degree. *Advances in Computational Mathematics*, 4 (1):389–396, Dec 1995. ISSN 1572-9044. doi: 10.1007/BF02123482. URL https://doi.org/10.1007/BF02123482.

E. Westenbrink, R.P. Arasaradnam, N. O'Connell, C. Bailey, C. Nwokolo, K.D. Bardhan, and J.A. Covington. Development and application of a new electronic nose instrument for the detection of colorectal cancer. *Biosensors and Bioelectronics*, 67:733 – 738, 2015. ISSN 0956-5663. Special Issue: BIOSENSORS 2014.

Ashley Wilks, Matthew Hart, Andrew Koehl, John Somerville, Billy Boyle, and David Ruiz-Alonso. Characterization of a miniature, ultra-high-field, ion mobility spectrometer. *International Journal for Ion Mobility Spectrometry*, 15(3):199–222, 2012. ISSN 1865-4584. doi: 10.1007/s12127-012-0109-x. URL http://dx.doi.org/10.1007/s12127-012-0109-x.

Alphus D. Wilson and Manuela Baietto. Advances in Electronic-Nose Technologies Developed for Biomedical Applications. *Sensors*, 11(1):1105–1176, 2011.

World Health Organization. *Global Tuberculosis Report 2016*. Global Tuberculosis Control. World Health Organization, 2016. ISBN 9789241565394. URL https://books.google.co.uk/books?id=Q8xRMQAACAAJ.

Hui Zou, Trevor Hastie, and Robert Tibshirani. Sparse principal component analysis. *Journal of Computational and Graphical Statistics*, 15(2):265–286, 2006. doi: 10.1198/106186006X113430. URL https://doi.org/10.1198/106186006X113430.